News

The Newsletter of the R Project

Volume 8/1, May 2008

Editorial

by John Fox

Welcome to the first issue of *R News* for 2008. The publication of this issue follows the recent release of R version 2.7.0, which includes a number of enhancements to R, especially in the area of graphics devices. Along with changes to the new version of R, Kurt Hornik reports on changes to CRAN, which now comprises more than 1400 contributed packages, and the Bioconductor Team reports news from the Bioconductor Project. This issue also includes news from the R Foundation, and an announcement of the 2008 *useR!* conference to be held in July in Dortmund, Germany.

R News depends upon the outstanding service of external reviewers, to whom the editorial board is deeply grateful. Individuals serving as referees during 2007 are acknowledged at the end of the current issue.

This issue features a number of articles that should be of interest to users of R: Gregor Gorjanc explains how `Sweave`, a literate-programming facility that is part of the standard R distribution, can be used with the LyX front-end to LaTeX. Jeff Enos and his co-authors introduce the **tradeCosts** package, which implements computations and graphs for securities transactions. Hormuzd Katki and Steven Mark describe the **NestedCohort** package for fitting standard survival models, such as the Cox model, when some information on covariates is missing. Vito Muggeo presents the **segmented** package for fitting piecewise-linear regression models. Cathy Chen and her colleagues show how the **BAYSTAR** package is used to fit two-regime threshold autoregressive (TAR) models using Markov-chain Monte-Carlo methods. Vincent Goulet and Matthieu Pigeon introduce the **actuar** package, which adds actuarial functionality to R.

The current issue of *R News* also marks the revival of two columns: Robin Hankin has contributed a *Programmer's Niche* column that describes the implementation of multivariate polynomials in the **multipol** package; and, in a *Help Desk* column, Uwe Ligges and I explain the ins and outs of avoiding — and using — iteration in R.

Taken as a whole, these articles demonstrate the vitality and diversity of the R Project. The continued vitality of *R News*, however, depends upon contributions from readers, particularly from package developers. One way to get *your* package noticed among the 1400 on CRAN is to write about it in *R News*. Keep those contributions coming!

John Fox
McMaster University
Canada
`John.Fox@R-project.org`

Contents of this issue:

Using Sweave with LyX

How to lower the LaTeX/Sweave learning curve

by Gregor Gorjanc

Introduction

LaTeX (LaTeX Project, 2005) is a powerful typesetting language, but some people find that acquiring a knowledge of LaTeX presents a steep learning curve in comparison to other "document processors." Unfortunately this extends also to "tools" that rely on LaTeX. Such an example is Sweave (Leisch, 2002), which combines the power of R and LaTeX using literate programming as implemented in noweb (Ramsey, 2006). Literate programming is a methodology of combining program code and documentation in one (source) file. In the case of Sweave, the source file can be seen as a LaTeX file with parts (chunks) of R code. The primary goal of Sweave is not documenting the R code, but delivering results of a data analysis. LaTeX is used to write the text, while R code is replaced with its results during the process of compiling the document. Therefore, Sweave is in fact a literate reporting tool. Sweave is of considerable value, but its use is somewhat hindered by the steep learning curve needed to acquire LaTeX.

The R package **odfWeave** (Kuhn, 2006) uses the same principle as Sweave, but instead of LaTeX uses an XML-based markup language named Open Document Format (ODF). This format can be easily edited in OpenOffice. Although it seems that **odfWeave** solves problems for non-LaTeX users, LaTeX has qualities superior to those of OpenOffice. However, the gap is getting narrower with tools like OOoLaTeX (Piroux, 2005), an OpenOffice macro for writing LaTeX equations in OpenOffice, and Writer 2 LaTeX (Just, 2006), which provides the possibility of converting OpenOffice documents to LaTeX. LaTeX has existed for decades and it appears it will remain in use. Anything that helps us to acquire and/or use LaTeX is therefore welcome. LyX (LyX Project, 2006) definitely is such tool.

LyX is an open source document preparation system that works with LaTeX and other "companion" tools. In short, I see LyX as a "Word"-like WYSIWYM (What You See Is What You Mean) front-end for editing LaTeX files, with excellent import and export facilities. Manuals shipped with LyX and posted on the wiki site (`http://wiki.lyx.org`) give an accessible and detailed description of LyX, as well as pointers to LaTeX documentation. I heartily recommend these resources for studying LyX and LaTeX. Additionally, LyX runs on Unix-like systems, including MacOSX, as well as on MS Windows. The LyX installer for MS Windows provides a neat way to install all the tools that are needed to work with LaTeX in general. This is not a problem for GNU/Linux distributions since package management tools take care of the dependencies. TeX Live (TeX Live Project, 2006) is another way to get LaTeX and accompanying tools for Unix, MacOSX, and MS Windows. LyX is an ideal tool for those who may struggle with LaTeX, and it would be an advantage if it could also be used for Sweave. Johnson (2006) was the first to embark on this initiative. I have followed his idea and extended his work using recent developments in R and LyX.

In the following paragraphs I give a short tutorial "LyX & Sweave in action", where I also show a way to facilitate the learning of LaTeX and consequently of Sweave. The section "LyX customisation" shows how to customise LyX to work with Sweave. I close with some discussion.

LyX & Sweave in action

In this section I give a brief tutorial on using Sweave with LyX. You might also read the "Introduction to LyX" and "The LyX Tutorial" manuals for additional information on the first steps with LyX. In order to actively follow this tutorial you have to customise LyX as described in the section "LyX customisation".

Open LyX, create a new file with the `File -> New` menu and save it. Start typing some text. You can preview your work in a PDF via the `View -> PDF (*)` menu, where * indicates one of the tools/routes (`latex`, `pdflatex`, etc.) that are used to convert LaTeX file to PDF. The availability of different routes of conversion, as well as some other commands, depend on the availability of converters on your computer.

The literate document class

To enable literate programming with R you need to choose a document class that supports this methodology. Follow the `Document -> Settings` menu and choose one of the document classes that indicates Sweave, say "article (Sweave noweb)". That is all. You can continue typing some text.

Code chunk

To enter R code you have to choose an appropriate style so that LyX will recognise this as program code. R code typed with a standard style will be treated as standard text. Click on the button "Standard" (Figure 1 — top left) and choose a scrap style, which is used for program code (chunks) in literate programming documents. You will notice that now the text you type has a different colour (Figure 1). This is an indicator that you are in a paragraph with a scrap style. There are different implementations of literate

programming. Sweave uses a noweb-like implementation, where the start of a code chunk is indicated with <<>>=, while a line with @ in the first column indicates the end of a code chunk (Figure 1). Try entering:

```
<<myFirstChunkInLyX>>=
xObs <- 100; xMean <- 10; xVar <- 9
x <- rnorm(n=xObs, mean=xMean, sd=sqrt(xVar))
mean(x)
@
```

Did you encounter any problems after hitting the ENTER key? LyX tries to be restrictive with spaces and new lines. A new line always starts a new paragraph with a standard style. To keep the code "together" in one paragraph of a scrap style, you have to use CTRL+ENTER to go onto a new line. You will notice a special symbol (Figure 1) at the end of the lines marking unbroken newline. Now write the above chunk of R code, save the file and preview a PDF. If the PDF is not shown, check the customisation part or read further about errors in code chunks. You can use all the code chunk options in the <<>>= markup part. For example <<echo=FALSE, fig=TRUE>>=, will have an effect of hidding output from R functions, while plots will be produced and displayed.

Inline code chunks

LyX also supports the inclusion of plain LaTeX code. Follow the Insert –> TeX Code menu, or just type CTRL+L and you will get a so-called ERT box (Figure 1) where you can type LaTeX code directly. This can be used for an inline code chunk. Create a new paragraph, type some text and insert \Sexpr{xObs} into the ERT box. Save the file and check the result in a PDF format. This feature can also be used for \SweaveOpts{} directives anywhere in the document. For example, \SweaveOpts{echo=FALSE} will suppress output from all R functions after that line. ERT boxes are advantageous since you can start using some LaTeX directly, but you can still produce whole documents without knowing the rest of the LaTeX commands that LyX has used.

Equations

Typing mathematics is one of the greatest strengths of LaTeX. To start an equation in LyX follow the Insert –> Math –> Inline/Display Formula menu or use CTRL+M and you will get an equation box. There is also a maths panel to facilitate the typing of symbols. You can also type standard LaTeX commands into the equation box and, say, \alpha will be automatically replaced with α. You can also directly include an inline code chunk in an equation, but note that backslash in front of Sexpr will not be displayed as can be seen in Figure 1.

Floats

A figure float can be filled with a code chunk and Sweave will replace the code chunk "with figures". How can we do this with LyX? Follow the Insert –> Float –> Figure menu and you will create a new box — a figure float. Type a caption and press the ENTER key. Choose the scrap style, insert the code chunk provided below (do not forget to use CTRL+ENTER), save the file, and preview in PDF format.

```
<<mySecondChunkInLyX, fig=TRUE>>=
hist(x)
@
```

If you want to center the figure, point the cursor at the code chunk, follow the Edit –> Paragraph Setting menu and choose alignment. This will center the code and consequently also the resulting figure. Alignment works only in LyX version 1.4.4 and later. You will receive an error with LyX version 1.4.3. If you still have LyX version 1.4.3, you can bypass this problem by retaining the default (left) alignment and by inserting LaTeX code for centering within a float, say \begin{center} above and \end{center} below the code chunk. Check the section "LyX customisation" for a file with such an example.

Errors in code chunks

If there are any errors in code chunks, the compiation will fail. LyX will only report that an error has occurred. This is not optimal as you never know where the error occured. There is a Python script listerrors shipped with LyX for this issue. Unfortunately, I do not know how to write an additional function for collecting errors from the R CMD Sweave process. I will be very pleased if anyone is willing to attempt this. In the meantime you can monitor the weaving process if you start LyX from a terminal. The weaving process will be displayed in a terminal as if R CMD Sweave is used (Figure 1, bottom right) and you can easily spot the problematic chunk.

Import/Export

You can import Sweave files into LyX via the File –> Import –> Sweave... menu. Export from LyX to Sweave and to other formats also works similarly. If you want to extract the R code from the document — i.e., tangle the document — just export to R/S code. Exported files can be a great source for studying LaTeX. However, this can be tedious, and I find that the View menu provides a handy way to examine LaTeX source directly. Preview of LaTeX and Sweave formats will work only if you set up a viewer/editor in the 'preferences' file (Figure 3) as shown in the following section. Do something in LyX and take a look at the produced LaTeX file via the View menu. This way you can easily become acquainted with LaTeX.

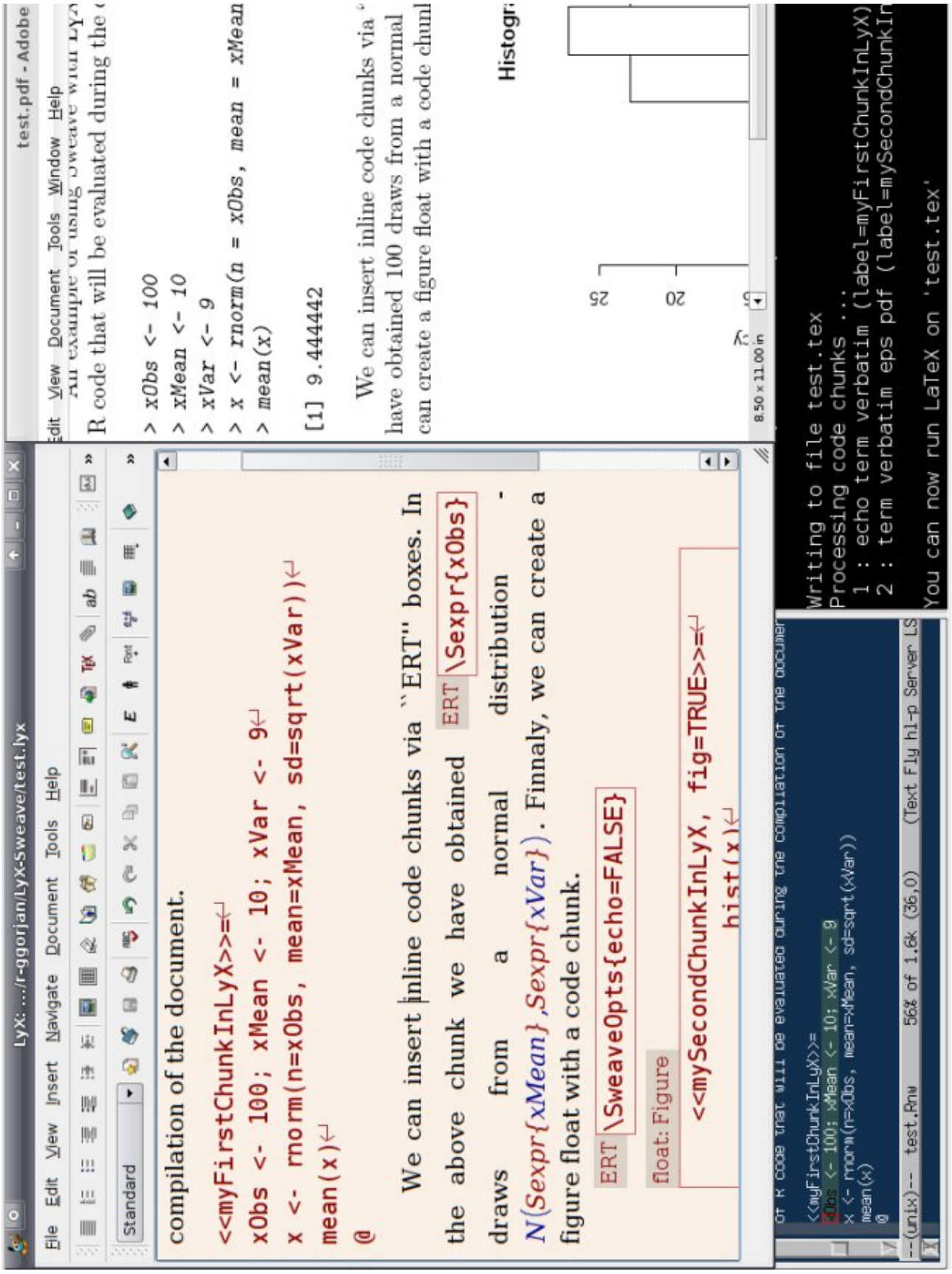

Figure 1: Screenshot of LyX with `Sweave` in action: LyX GUI (top-left), produced PDF (top-right), source code (Sweave) in an editor (bottom-left), and echo from weaving in a terminal (bottom-right)

In LyX version 1.5 the user can monitor LaTeX code instantly in a separate window. Users of LyX can therefore easily become acquainted with LaTeX and there should be even less reason not to use Sweave.

LyX customisation

LyX already supports noweb-like literate programming as described in the "Extended LyX Features" manual. Unfortunately, the default implementation does not work with R. To achieve this, LyX needs to be customised to use R for weaving (replacing R code with its ouput) and tangling (extracting program code), while LyX will take care of the conversion into the chosen output format, for example, PostScript, PDF, etc. LyX can convert to, as well as from, many formats, which is only a matter of having proper converters. For example latex is used to convert a LaTeX file to DVI format, dvips is used to convert a DVI file to PostScript, and you can easily deduce what the ps2pdf converter does. Of course, pdflatex can also be used to directly convert LaTeX to PDF. So, the idea of providing Sweave support to LyX is to specify a converter (weaver) of a Sweave file that will be used for the evaluation of R code, replacing it with the results in the generated LaTeX file. Additionally, a tangler needs to be specified if only the extraction of R code is required. I describe such customisation in this section, which is deliberately detailed so that anyone with interest and C++ experience could work with the LyX team on direct support of Sweave. I also discuss a possible way for this in the subsection "Future work".

Users can customise LyX via the Tools -> Preferences menu or via configuration files. Although menus may be more convenient to use, I find that handling a configuration file is easier, less cluttered and better for the customisation of LyX on different machines. Since the readers of this newsletter already know how to work with R code, the handling of another ASCII file will not present any problems. The use of menus in LyX should be obvious from the given description. Configuration files for LyX can be saved in two places: the so-called library and the user directory. As usual, the settings in the user directory take precedence over those in the library directory and I will show only the customisation for the user. The manual "Customizing LyX: Features for the Advanced User" describes all LyX customisation features as well as system-wide customisation. The configuration file in the user directory is named 'preferences'. **Formats**, **converters**, and **document classes** need to be customised to enable Sweave support in LyX. I will describe each of these in turn. Skip to the subsection "Install" on page 7, if you are not interested in the details.

Formats

LyX formats describe general information about file formats. The default specification for the LaTeX file format is shown in Figure 2. This specification consists of the following fields:

- format name ("latex");
- file extension ("tex");
- format name that is displayed in the LyX GUI ("Latex (Plain)");
- keyboard shortcut ("L");
- viewer name ("");
- editor name ("");
- type of the document and vector graphics support by the document ("document").

Literate programming in LyX is implemented via the literate file format. The latter needs to be modified to work with R, and a new file format for R code must be introduced. The name literate must be used as this is a special file format name in LyX for literate programming based on the noweb implementation. The entries in the 'preferences' file for a modified literate file format and a new r file format are shown in Figure 3. The values in separate fields are more or less obvious — editor stands for your favourite editor such as Emacs, Kate, Notepad, Texmaker, Tinn-R, vi, WinEdt, Wordpad, etc. It is very useful to define your favourite editor for both the viewing and the editing of Sweave, R, latex, and pdflatex file formats. This provides the possibility of viewing the file in these formats from LyX with only two clicks, as noted in the "LyX & Sweave in action" section.

Converters

I have already mentioned that LyX has a powerful feature of converting between various file formats with the use of external converter tools. For our purpose, only tools to weave and tangle need to be specified, while LyX will take care of all other conversions. To have full support for Sweave in LyX the following conversions are required:

- convert (import) the Sweave file into a LyX file with R chunks;
- convert (weave) the LyX file with R chunks to a specified output format (LaTeX, PostScript, PDF, etc.);
- convert (tangle) the LyX file with R chunks to a file with R code only; and
- convert (export) LyX file with R chunks to a Sweave file.

```
\format "latex" "tex" "Latex (Plain)" "L" "" "" "document"
```

Figure 2: The default format specification for a LaTeX file

```
#
# FORMATS SECTION ##########################
#

\format "literate" "Rnw" "Sweave"           "" "editor" "editor" "document"
\format "r"        "R"   "R/S code"         "" "editor" "editor" "document"
\format "latex"    "tex" "LaTeX (plain)"    "" "editor" "editor" "document"
\format "pdflatex" "tex" "LaTeX (pdflatex)" "" "editor" "editor" "document"

#
# CONVERTERS SECTION ########################
#

\converter "literate" "r"        "R CMD Stangle $$i" ""
\converter "literate" "latex"    "R CMD Sweave $$i"  ""
\converter "literate" "pdflatex" "R CMD Sweave $$i"  ""
```

Figure 3: Format and converter definitions for Sweave support in LyX

The first task can be accomplished with LyX's import utility tool tex2lyx and its option -n to convert a literate programming file, in our case a Sweave file, to the LyX file format. This can be done either in a terminal "by hand" (tex2lyx -n file.Rnw) or via the File –> Import menu within LyX. No customisation is required for this task. tex2lyx converts the literate programming file to the LyX file format with two minor technicalities of which it is prudent to be aware. The first one is that LyX uses the term scrap instead of chunk. This is due to a historical reason and comes from another literate programming tool named nuweb (Briggs et al., 2002). I shall use both terms (scrap and chunk) interchangeably to refer to the part of the document that contains the program code. Another technicality is related to the \documentclass directive in a LaTeX/Sweave file. At the time of writing, LyX provides article, report and book LaTeX classes for literate programming. These are provided via document classes that will be described later on.

When converting a LyX file with R chunks to other formats, the information on how to weave and possibly also tangle the file is needed. The essential part of this task is the specification of R scripts Sweave and Stangle in a 'preferences' file as shown in Figure 3. These scripts are part of R from version 2.4.0. Note that two converters are defined for weaving: one for latex and one for the pdflatex file format. This way both routes of LaTeX conversion are supported — i.e., LaTeX –> PostScript –> PDF for the latex file format, and LaTeX –> PDF for the pdflatex file format. The details of weaving and tangling processes are described in the "Extended LyX Features" manual.

Document classes

LyX uses layouts for the definition of environments/styles, for example the standard layout/style for normal text and the scrap layout/style for program code in literate programming. Layout files are also used for the definition of document classes, sometimes also called text classes. Document classes with literate support for the article, report and book LaTeX document classes already exist. The definitions for these files can be found in the 'layout' subdirectory of the LyX library directory. The files are named 'literate-article.layout', 'literate-report.layout' and 'literate-book.layout'. That is the reason for the mandatory use of the literate file format name as described before in the formats subsection. All files include the 'literate-scrap.inc' file, where the scrap style is defined. The syntax of these files is simple and new files for other document classes can be created easily. When LyX imports a literate programming file it automatically chooses one of these document classes, based on a LaTeX document class.

The default document classes for literate programming in LyX were written with noweb in mind. There are two problems associated with this. The default literate document classes are available to the LyX user only if the 'noweb.sty' file can be found by LaTeX during the configuration of LyX — done during the first start of LyX or via the Tools –> Reconfigure menu within LyX. This is too restrictive for Sweave users, who require the 'Sweave.sty' file. Another problem is that the default literate class does not allow aligning the scrap style. This means that the R users cannot center figures.

To avoid the aforementioned problems, I provide modified literate document class files that provide a smoother integration of Sweave and LyX. The files have the same names as their "noweb" originals.

The user can insert R code into the Sweave file with noweb- like syntax

```
<<>>=
someRCode
@
```

or LaTeX-like syntax

```
\begin{Scode}
someRCode
\end{Scode}
```

or even a mixture of these two (Leisch, 2002). LyX could handle both types, but LyX's definition of the style of LaTeX-like syntax cannot be general enough to fulfil all the options Sweave provides. Therefore, only noweb-like syntax is supported in LyX. Nevertheless, it is possible to use LaTeX-like syntax, but one has to resort to the use of plain LaTeX markup.

LyX has been patched to incorporate the \SweaveSyntax{}, \SweaveOpts{}, \SweaveInput{}, \Sexpr{} and \Scoderef{} commands. These commands will be handled appropriately during the import of the Sweave file into LyX. The same holds for the LaTeX environment Scode, but the default layout in LyX used for this environment is not as useful as the noweb-like syntax.

"Install"

At least LyX version 1.4.4 and R version 2.4.0 are needed. All files ('preferences', 'literate-article.layout', 'literate-report.layout', 'literate-book.layout', and 'literature-scrap.inc') that are mentioned in this section are available at http://cran.r-project.org/contrib/extra/lyx. There are also other files ('test.lyx', 'Sweave-test-1.lyx', and 'template-vignette.lyx') that demonstrate the functionality. Follow these steps to enable use of Sweave in LyX:

- find the so-called LyX user directory via the Help –> About LyX menu within LyX;
- save the 'preferences' file in the LyX user directory;
- save the 'literate-*.*' files to the 'layouts' subdirectory of the LyX user directory;
- assure that LaTeX can find and use the 'Sweave.sty' file (read the TeX path system subsection if you have problems with this);
- start LyX and update the configuration via the Tools –> Reconfigure menu; and
- restart LyX.

It is also possible to use LyX version 1.4.3, but there are problems with the alignment of code chunk results in floats. Use corresponding files from the 'lyx-1.4.3' subdirectory at http://cran.r-project.org/contrib/extra/lyx. Additionally, save the 'syntax.sweave' file in the LyX user directory.

TeX path system

It is not the purpose of this article to describe LaTeX internals. However, R users who do not have experience with LaTeX (the intended readership) might encounter problems with the path system that LaTeX uses and I shall give a short description to overcome this. So far I have been referring to LaTeX, which is just a set of commands at a higher level than "plain" TeX. Both of them use the same path system. When you ask TeX to use a particular package (say Sweave with the command \usepackage{Sweave}), TeX searches for necessary files in TeX paths, also called texmf trees. These trees are huge collections of directories that contain various files: packages, fonts, etc. TeX searches files in these trees in the following order:

- the root texmf tree such as '/usr/share/texmf', 'c:/texmf' or 'c:/Program Files/TEX/texmf';
- the local texmf tree such as '/usr/share/local/texmf'; 'c:/localtexmf' or 'c:/Program Files/TEX/texmf-local'; and
- the personal texmf tree in your home directory,

where TEX is a directory of your TeX distribution such as MiKTeX (Schenk, 2006). R ships 'Sweave.sty' and other TeX related files within its own texmf tree in the 'pathToRInstallDirectory/share/texmf' directory. You have to add R's texmf tree to the TeX path, and there are various ways to achieve this. I believe that the easiest way is to follow these steps:

- create the 'tex/latex/R' sub-directory in the local texmf tree;
- copy the contents of the R texmf tree to the newly created directory;
- rebuild TeX's filename database with the command texhash (MiKTeX has also a menu option for this task); and
- check if TeX can find 'Sweave.sty' — use the command kpsewhich Sweave.sty or findtexmf Sweave.sty in a terminal.

Users of Unix-like systems can use a link instead of a sub-directory in a local texmf tree to ensure the latest version of R's texmf tree is used. Debian GNU/Linux and its derivatives, with R installed from official Debian packages, have this setup automatically. Additional details on the TeX

path system can be found at http://www.ctan.org/installationadvice/. Windows useRs might also be interested in notes about using MiKTEX with R for Windows at http://www.murdoch-sutherland.com/Rtools/miktex.html.

Future work

The customisation described above is not a difficult task (just six steps), but it would be desirable if LyX could support Sweave "out of the box". LyX has a convenient configuration feature that is conditional on availability of various third party programs and LaTeX files. Sweave support for editing could be configured if 'Sweave.sty' is found, while R would have to be available for conversions. To achieve this, only minor changes would be needed in the LyX source. I think that the easiest way would be to add another argument, say -ns, to the tex2lyx converter that would drive the conversion of the Sweave file to LyX as it is done for noweb files, except that the Sweave-specific layout of files would be chosen. Additionally, the format name would have to be changed from literate to avoid collision with noweb. Unfortunately, these changes require C++ skills that I do not have.

Discussion

LyX is not the only "document processor" with the ability to export to LaTeX. AbiWord, KWord, and OpenOffice are viable open source alternatives, while I am aware of only one proprietary alternative, Scientific WorkPlace (SWP) (MacKichan Software, Inc., 2005). Karlsson (2006) reviewed and compared SWP with LyX. His main conclusions were that both LyX and SWP are adequate, but could "learn" from each other. One of the advantages of SWP is the computer algebra system MuPAD (SciFace Software, Inc., 2004) that the user gets with SWP. LyX has some support for GNU Octave, Maxima, Mathematica and Maple, but I have not tested it. Now Sweave brings R and its packages to LyX, so the advantage of SWP in this regard is diminishing. Additionally, LyX and R (therefore also Sweave) run on all major platforms, whereas SWP is restricted to Windows.

Sweave by default creates PostScript and PDF files for figures. This eases the conversion to either PostScript and/or PDF of a whole document, which LyX can easily handle. The announced support for the PNG format (Leisch, personal communication) in Sweave will add the possibility to create lighter PDF files. Additionally, a direct conversion to HTML will be possible. This is a handy alternative to **R2HTML** (Lecoutre, 2003), if you already have a Sweave source.

The current default for R package-vignette files is Sweave, and since Sweave is based on LaTeX, some developers might find it hard to write vignettes. With LyX this need not be the case anymore, as vignettes can also be created with LyX. Developers just need to add vignette-specific markup, i.e., %\VignetteIndexEntry{}, %\VignetteDepends{}, %\VignetteKeywords{} and %\VignettePackage{}, to the document preamble via the Document -> Settings -> LaTeX Preamble menu within LyX. A template for a vignette (with vignette specific markup already added) is provided in the file 'template-vignette.lyx' at http://cran.r-project.org/contrib/extra/lyx. A modified layout for Sweave in LyX also defines common LaTeX markup often used in vignettes, for example, \Rcode{}, \Robject{}, \Rcommand{}, \Rfunction{}, \Rfunarg{}, \Rpackage{}, \Rmethod{}, and \Rclass{}.

Summary

I have shown that it is very easy to use LyX for literate programming/reporting and that the LaTeX/Sweave learning curve need not be too steep.

LyX does not support Sweave out of the box. I describe the needed customisation, which is very simple. I hope that someone with an interest will build upon the current implementation and work with the LyX developers on the direct support of Sweave.

Acknowledgements

I would like to thank the LyX team for developing such a great program and incorporating patches for smoother integration of LyX and Sweave. Acknowledgements go also to Friedrich Leisch for developing Sweave in the first place, as well as for discussion and comments. Inputs by John Fox have improved the paper.

Bibliography

P. Briggs, J. D. Ramsdell, and M. W. Mengel. Nuweb: *A Simple Literate Programming Tool*, 2002. URL http://nuweb.sourceforge.net. Version 1.0b1.

P. E. Johnson. How to use LyX with R, 2006. URL http://wiki.lyx.org/LyX/LyxWithRThroughSweave.

H. Just. *Writer* $\overrightarrow{2}$ *LaTeX*, 2006. URL http://www.hj-gym.dk/~hj/writer2latex. Version 0.4.1d.

A. Karlsson. Scientific workplace 5.5 and LyX 1.4.2. *Journal of Statistical Software*, 17(Software Review 1):1–11, 2006. URL http://www.jstatsoft.org/v17/s01/v17s01.pdf.

M. Kuhn. Sweave and the open document format – the odfWeave package. *R News*, 6(4):2–8, 2006. URL http://CRAN.R-project.org/doc/Rnews/Rnews_2006-4.pdf.

E. Lecoutre. The R2HTML package. *R News*, 3(3):33–36, 2003. URL http://CRAN.R-project.org/doc/Rnews/Rnews_2003-3.pdf.

F. Leisch. Dynamic generation of statistical reports using literate data analysis. In W. Haerdle and B. Roenz, editors, *Compstat 2002 - Proceedings in Computational Statistics*, pages 575–580, Heidelberg, Germany, 2002. Physika Verlag. ISBN 3-7908-1517-9.

TEX Live Project. *A distribution of TeX and friends*, 2006. URL http://www.tug.org/texlive/. Version 2005-11-01.

MacKichan Software, Inc. *Scientific Workplace*, 2005. URL http://www.mackichan.com. Version 5.5.

G. Piroux. *OOoLATEX*, 2005. URL http://ooolatex.sourceforge.net. Version 2005-10-19.

LATEX Project. *LATEX - A document preparation system*, 2005. URL http://www.latex-project.org/. Version 2005-12-01.

LYX Project. *LYX - The Document Processor*, 2006. URL http://www.lyx.org. Version 1.4.4.

N. Ramsey. Noweb - *a simple, extensible tool for literate programming*, 2006. URL http://www.eecs.harvard.edu/~nr/noweb. Version 2.11b.

C. Schenk. *MikTEX Project*, 2006. URL http://www.miktex.org/. Version 2.5.

SciFace Software, Inc. *MuPad*, 2004. URL http://www.sciface.com. Version 3.1.

Gregor Gorjanc
University of Ljubljana
Biotechnical faculty
Slovenia
gregor.gorjanc@bfro.uni-lj.si

Trade Costs

by Jeff Enos, David Kane, Arjun Ravi Narayan, Aaron Schwartz, Daniel Suo and Luyi Zhao

Introduction

Trade costs are the costs a trader must pay to implement a decision to buy or sell a security. Consider a single trade of a single equity security. Suppose on the evening of August 1, a trader decides to purchase 10,000 shares of IBM at \$10, the *decision price* of the trade. The next day, the trader's broker buys 10,000 shares in a rising market and pays \$11 per share, the trade's *execution price*.

How much did it cost to implement this trade? In the most basic ex-post analysis, trade costs are calculated by comparing the execution price of a trade to a benchmark price.[1] Suppose we wished to compare the execution price to the price of the security at the time of the decision in the above example. Since the trader's decision occurred at \$10 and the broker paid \$11, the cost of the trade relative to the decision price was \$11 − \$10 = \$1 per share, or \$10,000 (9.1% of the total value of the execution).

Measuring costs relative to a trade's decision price captures costs associated with the delay in the release of a trade into the market and movements in price after the decision was made but before the order is completed. It does not, however, provide a means to determine whether the broker's execution reflects a fair price. For example, the price of \$11 would be a poor price if most transactions in IBM on August 2 occurred at \$10.50. For this purpose a better benchmark would be the day's volume-weighted average price, or VWAP. If VWAP on August 2 was \$10.50 and the trader used this as her benchmark, then the trade cost would be \$0.50 per share, or \$5,000.

The first version of the **tradeCosts** package provides a simple framework for calculating the cost of trades relative to a benchmark price, such as VWAP or decision price, over multiple periods along with basic reporting and plotting facilities to analyse these costs.

Trade costs in a single period

Suppose we want to calculate trade costs for a single period. First, the data required to run the analysis must be assembled into three data frames.

The first data frame contains all trade-specific information, a sample of which is in the `trade.mar.2007` data frame:

```
> library("tradeCosts")
> data("trade.mar.2007")
> head(trade.mar.2007)
```

```
      period       id side exec.qty exec.price
1 2007-03-01 03818830    X    60600       1.60
2 2007-03-01 13959410    B     4400      32.21
3 2007-03-01 15976510    X    13600       7.19
4 2007-03-01 22122P10    X   119000       5.69
5 2007-03-01 25383010    X     9200       2.49
6 2007-03-01 32084110    B     3400      22.77
```

Trading data must include at least the set of columns included in the sample shown above: `period` is the (arbitrary) time interval during which the trade was executed, in this case a calendar trade day; `id` is a unique security identifier; `side` must be one of `B` (buy), `S` (sell), `C` (cover) or `X` (short sell); `exec.qty` is the number of shares executed; and `exec.price` is the price per share of the execution. The `create.trade.data` function can be used to create a data frame with all of the necessary information.

Second, trade cost analysis requires dynamic descriptive data, or data that changes across periods for each security.

```
> data("dynamic.mar.2007")
> head(dynamic.mar.2007[c("period", "id", "vwap",
+     "prior.close")])
```

```
      period       id   vwap prior.close
1 2007-03-01 00797520   3.88        3.34
2 2007-03-01   010015 129.35        2.53
3 2007-03-01   023282 613.57       12.02
4 2007-03-01 03818830   1.58        1.62
5 2007-03-01   047628 285.67        5.61
6 2007-03-01   091139 418.48        8.22
```

The `period` and `id` columns match those in the trading data. The remaining two columns in the sample are benchmark prices: `vwap` is the volume-weighted average price for the period and `prior.close` is the security's price at the end of the prior period.

The third data frame contains static data for each security.

```
> data("static.mar.2007")
> head(static.mar.2007)
```

```
           id symbol                          name sector
1301 00036020   AAON                     Aaon Inc    IND
2679 00036110    AIR                     Aar Corp    IND
3862 00040010   ABCB               Ameris Bancorp    FIN
406  00080S10   ABXA                  Abx Air Inc    IND
3239 00081T10    ABD            Acco Brands Corp    IND
325  00083310    ACA Aca Capital Hldgs Inc -redh    FIN
```

The `id` column specifies an identifier that can be linked to the other data frames. Because this data is static, there is no `period` column.

Once assembled, these three data frames can be analysed by the `trade.costs` function:

[1] For an in-depth discussion of both ex-ante modeling and ex-post measurement of trade costs, see Kissell and Glantz (2003).

```
> result <- trade.costs(trade = trade.mar.2007,
+     dynamic = dynamic.mar.2007,
+     static = static.mar.2007,
+     start.period = as.Date("2007-03-01"),
+     end.period = as.Date("2007-03-01"),
+     benchmark.price = "vwap")
```

The trade, dynamic, and static arguments refer to the three data frames discussed above. start.period and end.period specify the period range to analyse. This example analyses only one period, March 1, 2007, and uses the vwap column of the dynamic data frame as the benchmark price. result is an object of class tradeCostsResults.

```
> summary(result)

Trade Cost Analysis

Benchmark Price: vwap

Summary statistics:
 Total Market Value:         1,283,963
 First Period:              2007-03-01
 Last Period:               2007-03-01
 Total Cost:                    -6,491
 Total Cost (bps):                 -51

Best and worst batches over all periods:
                            batch.name exec.qty   cost
1 22122P10 (2007-03-01 - 2007-03-01)  119,000 -3,572
2 03818830 (2007-03-01 - 2007-03-01)   60,600 -1,615
3 88362320 (2007-03-01 - 2007-03-01)   31,400 -1,235
6 25383010 (2007-03-01 - 2007-03-01)    9,200     33
7 13959410 (2007-03-01 - 2007-03-01)    4,400    221
8 32084110 (2007-03-01 - 2007-03-01)    3,400    370

Best and worst securities over all periods:
        id exec.qty   cost
1 22122P10  119,000 -3,572
2 03818830   60,600 -1,615
3 88362320   31,400 -1,235
6 25383010    9,200     33
7 13959410    4,400    221
8 32084110    3,400    370

NA report:
           count
id             0
period         0
side           2
exec.price     0
exec.qty       0
vwap           1
```

The first section of the report provides high-level summary information. The total unsigned market value of trades for March 1 was around $1.3 million. Relative to VWAP, these trades cost -$6,491, indicating that overall the trades were executed at a level "better" than VWAP, where better buys/covers (sells/shorts) occur at prices below (above) VWAP. This total cost is the sum of the signed cost of each trade relative to the benchmark price. As a percentage of total executed market value, this set of trades cost -51 bps relative to VWAP.

The next section displays the best and worst *batches* over all periods. We will discuss batches in the next section. For now, note that when dealing with only one period, each trade falls into its own batch, so this section shows the most and least expensive trades for March 1. The next section displays the best and worst securities by total cost across all periods. Because there is only one trade per security on March 1, these results match the best and worst batches by cost.

Calculating the cost of a trade requires a non-NA value for id, period, side, exec.price, exec.qty and the benchmark price. The final section shows a count for each type of NA in the input data. Rows in the input data with NA's in any of these columns are removed before the analysis is performed and reported here.

Costs over multiple periods

Calculating trade costs over multiple periods works similarly. Cost can be calculated for each trade relative to a benchmark price which either varies over the period of the trade or is fixed at the decision price.

Suppose, for example, that the trader decided to short a stock on a particular day, but he wanted to trade so many shares that it took several days to complete the order. For instance, consider the following sequence of trades in our sample data set for Progressive Gaming, PGIC, which has id 59862K10:

```
> subset(trade.mar.2007, id %in% "59862K10")

        period       id side exec.qty exec.price
166 2007-03-13 59862K10    X    31700        5.77
184 2007-03-15 59862K10    X    45100        5.28
218 2007-03-19 59862K10    X   135800        5.05
259 2007-03-20 59862K10    X    22600        5.08
```

How should we calculate the cost of these trades? We could calculate the cost for each trade separately relative to a benchmark price such as vwap, exactly as in the last example. In this case, the cost of each trade in PGIC would be calculated relative to VWAP in each period and then added together. However, this method would ignore the cost associated with spreading out the sale over several days. If the price of the stock had been falling over the four days of the sale, for example, successive trades appear less attractive when compared to the price at the time of the decision. The trader can capture this cost by grouping the four short sales into a *batch* and comparing the execution price of each trade to the batch's original decision price.

Performing this type of multi-period analysis using **tradeCosts** requires several modifications to the previous single period example. Note that since no period range is given, analysis is performed over the entire data set:

```
> result.batched <- trade.costs(trade.mar.2007,
+     dynamic = dynamic.mar.2007,
+     static = static.mar.2007,
+     batch.method = "same.sided",
+     benchmark.price = "decision.price")
```

First, `trade.costs` must be instructed how to group trades into batches by setting the `batch.method` parameter. This version of **tradeCosts** provides a single multi-period sample batch method, `same.sided`, which groups all consecutive same-sided orders into a single batch. Provided there were no buys in between the four sales in PGIC, all four trades would be grouped into the same batch. Second, setting `benchmark.price` to `decision.price` sets the benchmark price to the prior closing price of the first trade in the batch. Running `summary` on the new result yields the following:

```
> summary(result.batched)

Trade Cost Analysis

Benchmark Price: decision.price

Summary statistics:
 Total Market Value:        47,928,402
 First Period:              2007-03-01
 Last Period:               2007-03-30
 Total Cost:                   587,148
 Total Cost (bps):                 123

Best and worst batches over all periods:
                           batch.name exec.qty    cost
1   04743910 (2007-03-19 - 2007-03-19)   17,800 -82,491
2   31659U30 (2007-03-09 - 2007-03-13)   39,800 -33,910
3   45885A30 (2007-03-13 - 2007-03-19)  152,933 -31,904
274 49330810 (2007-03-13 - 2007-03-30)   83,533  56,598
275 15649210 (2007-03-15 - 2007-03-28)   96,900  71,805
276 59862K10 (2007-03-13 - 2007-03-20)  235,200 182,707

Best and worst securities over all periods:
          id exec.qty    cost
1   04743910   17,800 -82,491
2   31659U30   51,400 -32,616
3   45885A30  152,933 -31,904
251 49330810   83,533  56,598
252 15649210  118,100  73,559
253 59862K10  235,200 182,707

NA report:
            count
id              0
period          0
side            6
exec.price      0
exec.qty        0
prior.close     2
```

This analysis covers almost $50 million of executions from March 1 to March 30, 2007. Relative to decision price, the trades cost $587,148, or 1.23% of the total executed market value.

The most expensive batch in the result contained the four sells in PGIC (`59862K10`) from March 13 to March 20, which cost $182,707.

Plotting results

The **tradeCosts** package includes a `plot` method that displays bar charts of trade costs. It requires two arguments, a `tradeCostsResults` object, and a character string that describes the type of plot to create.

The simplest plot is a time series of total trade costs in basis points over each period:

```
> plot(result.batched, "time.series.bps")
```

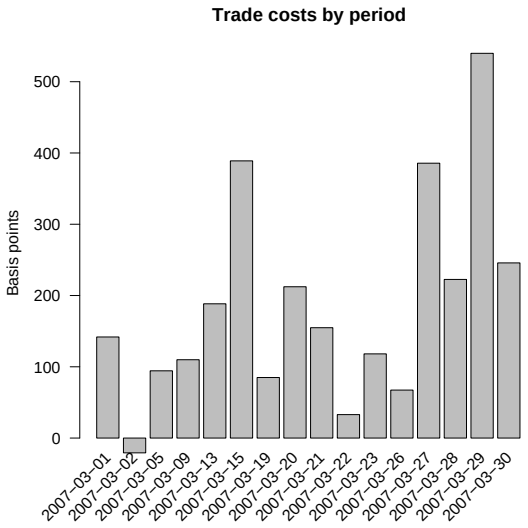

Figure 1: A time series plot of trade costs.

This chart displays the cost for each day in the previous example. According to this chart, all days had positive cost except March 2.

The second plot displays trade costs divided into categories defined by a column in the `static` data frame passed to `trade.costs`. Since `sector` was a column of that data frame, we can look at costs by company sector:

```
> plot(result.batched, "sector")
```

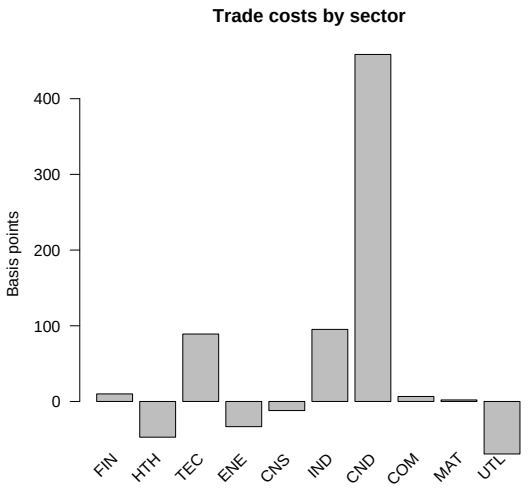

Figure 2: A plot of trade costs by sector.

Over the period of the analysis, trades in `CND` were especially expensive relative to decision price.

The last plot applies only to `same.sided` batched trade cost analysis as we performed in the multi-period example. This chart shows cost separated into the different periods of a batch. The cost of the first batch of PGIC, for example, contributes to the first bar, the cost of the second batch to the second bar, and so on.

```
> plot(result.batched, "cumulative")
```

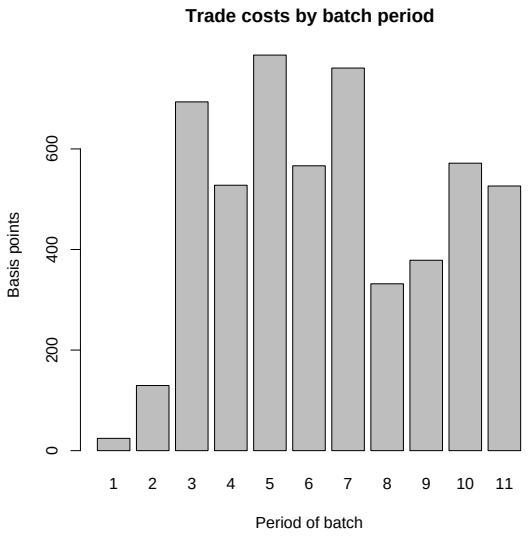

Figure 3: Costs by batch period, in bps.

As one might expect, the first and second trades in a batch are the cheapest with respect to decision price because they occur closest to the time of the decision.

Conclusion

tradeCosts currently provides a simple means of calculating the cost of trades relative to a benchmark price over multiple periods. Costs may be calculated relative to a period-specific benchmark price or, for trades spanning multiple periods, the initial decision price of the trade. We hope that over time and through collaboration the package will be able to tackle more complex issues, such as ex-ante modeling and finer compartmentalization of trade costs.

Bibliography

R. Kissell and M. Glantz. *Optimal Trading Strategies*. American Management Association, 2003.

Jeff Enos, David Kane, Arjun Ravi Narayan, Aaron Schwartz, Daniel Suo, Luyi Zhao
Kane Capital Management
Cambridge, Massachusetts, USA
`jeff@kanecap.com`

Survival Analysis for Cohorts with Missing Covariate Information

by Hormuzd A. Katki and Steven D. Mark

NestedCohort fits Kaplan-Meier and Cox Models to estimate standardized survival and attributable risk for studies where covariates of interest are observed on only a sample of the cohort. Missingness can be either by happenstance or by design (for example, the case-cohort and case-control within cohort designs).

Introduction

Most large cohort studies have observations with missing values for one or more exposures of interest. Exposure covariates that are missing by chance (missing by happenstance) present challenges in estimation well-known to statisticians. Perhaps less known is that most large cohort studies now include analyses of studies which deliberately sample only a subset of all subjects for the measurement of some exposures. These "missingness by design" studies are used when an exposure of interest is expensive or difficult to measure. Examples of different sampling schemes that are used in missing by design studies are the case-cohort, nested case-control, and case-control studies nested within cohorts in general (Mark and Katki (2001); Mark (2003); Mark and Katki (2006)). Missingness by design can yield important cost savings with little sacrifice of statistical efficiency (Mark (2003); Mark and Katki (2006)). Although for missingness-by-happenstance, the causes of missingness are not controlled by the investigator, the validity of any analysis of data with missing values depends on the relationship between the observed data and the missing data. Except under the strongest assumption that missing values occur completely at random (MCAR), standard estimators that work for data without missing values are biased when used to analyze data with missing values.

Mark (2003); Mark and Katki (2006) propose a class of weighted survival estimators that accounts for either type of missingness. The estimating equations in this class weight the contribution from completely observed subjects by the inverse probability of being completely observed (see below), and subtract an 'offset' to gain efficiency (see above references). The probabilities of being completely observed are estimated from a logistic regression. The predictors for this logistic regression are some (possibly improper) subset of the covariates for which there are no missing values; the outcome is an indicator variable denoting whether each observation has measurements for all covariates. The predictors may include the outcome variables (time-to-event), exposure variables that are measured on all subjects, and any other variables measured on the entire cohort. We refer to variables that are neither the outcome, nor in the set of exposures of interest (e.g. any variable used in the estimation of the Cox model), as auxiliary variables.

The weighted estimators we propose are unbiased when the missing mechanism is missing-at-random (MAR) and the logistic regression is correctly specified. For missing-by-design, MAR is satisfied and the correct logistic model is known. If there is any missing-by-happenstance, MAR is unverifiable. Given MAR is true, a logistic model saturated in the completely-observed covariates will always be correctly specified. In practice, given that the outcome is continuous (time-to-event), fitting saturated models is not feasible. However, fitting as rich a model as is reasonably possible not only bolsters the user's case that the model is correctly specified, but also improves efficiency (Mark (2003); Mark and Katki (2006)). Also, auxiliary variables can produce impressive efficiency gains and hence should be included as predictors even when not required for correct model specification (Mark (2003); Mark and Katki (2006)).

Our R package **NestedCohort** implements much of the methodology of Mark (2003); Mark and Katki (2006). The major exception is that it does not currently implement the finely-matched nested case-control design as presented in appendix D of Mark (2003); frequency-matching, or no matching, in a case-control design are implemented. In particular, **NestedCohort**

1. estimates not just relative risks, but also absolute and attributable risks. **NestedCohort** can estimate both non-parametric (Kaplan-Meier) and semi-parametric (Cox model) survival curves for each level of the exposures also attributable risks that are standardized for confounders.

2. allows cases to have missing exposures. Standard nested case-control and case-cohort software can produce biased estimates if cases are missing exposures.

3. produces unbiased estimates when the sampling is stratified on any completely observed variable, including failure time.

4. extracts efficiency out of auxiliary variables available on all cohort members.

5. uses weights *estimated* from a correctly-specified sampling model to greatly increase the efficency of the risk estimates compared to using the 'true' weights (Mark (2003); Mark and Katki (2006)).

6. estimates relative, absolute, and attributable risks for vectors of exposures. For relative risks, any covariate can be continuous or categorical.

NestedCohort has three functions that we demonstrate in this article.

1. `nested.km`: Estimates the Kaplan-Meier survival curve for each level of categorical exposures.

2. `nested.coxph`: Fits the Cox model to estimate relative risks. All covariates and exposures can be continuous or categorical.

3. `nested.stdsurv`: Fits the Cox model to estimate standardized survival probabilities, and Population Attributable Risk (PAR). All covariates and exposures must be categorical.

Example study nested in a cohort

In Mark and Katki (2006), we use our weighted estimators to analyze data on the association of *H.Pylori* with gastric cancer and provide simulations that demonstrate the increases in efficiency due to using estimated weights and auxiliary variables. In this document, we present a second example. Abnet et al. (2005) observe esophageal cancer survival outcomes and relevant confounders on the entire cohort. We are interested in the effect of concentrations of various metals, especially zinc, on esophageal cancer. However, measuring metal concentrations consumes precious esophageal biopsy tissue and requires a costly measurement technique. Thus we measured concentrations of zinc (as well as iron, nickel, copper, calcium, and sulphur) on a sample of the cohort. This sample oversampled the cases and those with advanced baseline histologies (i.e. those most likely to become cases) since these are the most informative subjects. Due to cost and availability constraints, less than 30% of the cohort was sampled. For this example, **NestedCohort** will provide adjusted hazard ratios, standardized survival probabilities, and PAR for the effect of zinc on esophageal cancer.

Specifying the sampling model

Abnet et al. (2005) used a two-phase sampling design to estimate the association of zinc concentration with the development of esophageal cancer. Sampling probabilities were determined by case-control status and severity of baseline esophageal histology. The sampling frequencies are given in the table below:

Baseline Histology	Case	Control	Total
Normal	14 / 22	17 / 221	31 / 243
Esophagitis	19 / 26	22 / 82	41 / 108
Mild Dysplasia	12 / 17	19 / 35	31 / 52
Moderate Dysplasia	3 / 7	4 / 6	7 / 13
Severe Dysplasia	5 / 6	3 / 4	8 / 10
Carcinoma In Situ	2 / 2	0 / 0	2 / 2
Unknown	1 / 1	2 / 2	3 / 3
Total	56 / 81	67 / 350	123 / 431

The column "baseline histology" contains, in order of severity, classification of pre-cancerous lesions. For each cell, the number to the right of the slash is the total cohort members in that cell, the left is the number we sampled to have zinc observed (i.e. in the top left cell, we measured zinc on 14 of the 22 members who became cases and had normal histology at baseline). Note that for each histology, we sampled roughly 1:1 cases to controls (frequency matching), and we oversampled the more severe histologies (who are more informative since they are more likely to become cases). Thirty percent of the cases could not be sampled due to sample availability constraints.

Since the sampling depended on case/control status (variable `ec01`) crossed with the seven baseline histologies (variable `basehist`), this sampling scheme will be accounted for by each function with the statement '`samplingmod="ec01*basehist"`'. This allows each of the 14 sampling strata its own sampling fraction, thus reproducing the sampling frequencies in the table.

NestedCohort requires that each observation have nonzero sampling probability. For this table, each of the 13 non-empty strata must have have someone sampled in it. Also, the estimators require that there are no missing values in any variable in the sampling model. However, if there is missingness, for convenience, **NestedCohort** will remove from the cohort any observations that have missingness in the sampling variables and will print a warning to the user. There should not be too many such observations.

Kaplan-Meier curves

To make non-parametric (Kaplan-Meier) survival curves by quartile of zinc level, use `nested.km`. These Kaplan-Meier curves have the usual interpretation: they do not standardize for other variables, and do not account for competing risks.

To use this, provide both a legal formula as per the `survfit` function and also a sampling model to calculate stratum-specific sampling fractions. Note that the '`survfitformula`' and '`samplingmod`' require their arguments to be inside double quotes. The

'data' argument is required: the user must provide the data frame within which all variables reside in. This outputs the Kaplan-Meier curves into a survfit object, so all the methods that are already there to manipulate survfit objects can be used[1].

To examine survival from cancer within each quartile of zinc, allowing different sampling probabilities for each of the 14 strata above, use nested.km, which prints out a table of risk differences versus the level named in 'exposureofinterest'; in this case, it's towards "Q4" which labels the 4th quartile of zinc concentration:

```
> library(NestedCohort)
> mod <- nested.km(survfitformula =
+          "Surv(futime01,ec01==1)~znquartiles",
+          samplingmod = "ec01*basehist",
+          exposureofinterest = "Q4", data = zinc)

Risk Differences vs. znquartiles=Q4 by time 5893
        Risk Difference  StdErr       95% CI
Q4 - Q1         0.28175 0.10416  0.07760 0.4859
Q4 - Q2         0.05551 0.07566 -0.09278 0.2038
Q4 - Q3         0.10681 0.08074 -0.05143 0.2651

> summary(mod)
[...]

308 observations deleted due to missing
                znquartiles=Q1
 time n.risk n.event survival std.err    95% CI
  163  125.5    1.37    0.989  0.0108 0.925 0.998
 1003  120.4    1.57    0.976  0.0169 0.906 0.994
 1036  118.8    1.00    0.968  0.0191 0.899 0.990
[...]

                znquartiles=Q2
 time n.risk n.event survival std.err    95% CI
 1038  116.9    1.57    0.987  0.0133 0.909 0.998
 1064  115.3    4.51    0.949  0.0260 0.864 0.981
 1070  110.8    2.33    0.929  0.0324 0.830 0.971
[...]
```

summary gives the lifetable. Although summary prints how many observations were 'deleted' because of missing exposures, the 'deleted' observations still contribute to the final estimates via estimation of the sampling probabilities. Note that the lifetable contains the weighted numbers of those at risk and who had the developed cancer.

The option 'outputsamplingmod' returns the sampling model that the sampling probabilities were calculated from. Examine this model if warned that it didn't converge. If 'outputsamplingmod' is TRUE, then nested.km will output a list with 2 components, the survmod component being the Kaplan-Meier survfit object, and the other samplingmod component being the sampling model.

Plotting Kaplan-Meier curves

Make Kaplan-Meier plots with the plot function for survfit objects. All plot options for survfit objects can be used.

```
> plot(mod,ymin=.6,xlab="time",ylab="survival",
+      main="Survival by Quartile of Zinc",
+      legend.text=c("Q1","Q2","Q3","Q4"),
+      lty=1:4,legend.pos=c(2000,.7))
```

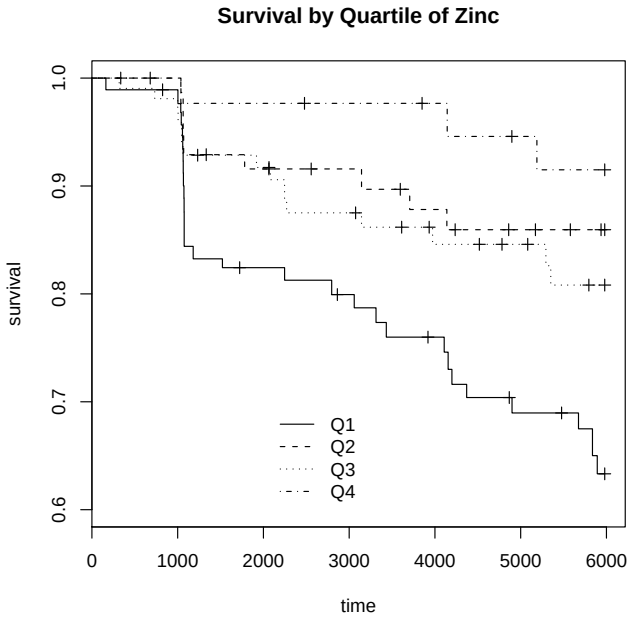

Figure 1: Kaplan-Meier plots by nested.km().

nested.km has some restrictions:

1. All variables are in a dataframe denoted by the 'data' argument.
2. No variable in the dataframe can be named o.b.s.e.r.v.e.d. or p.i.h.a.t.
3. 'survfitformula' must be a valid formula for survfit objects: All variables must be factors.
4. It does not support staggered entry into the cohort. The survival estimates will be correct, but their standard errors will be wrong.

Cox models: relative risks

To fit the Cox model, use nested.coxph. This function relies on coxph that is already in the survival package, and imitates its syntax as much as possible. In this example, we are interested in estimating the effect of zinc (as zncent, a continuous variable standardized to 0 median and where a 1 unit change is an

[1] nested.km uses the weights option in survfit to estimate the survival curve. However, the standard errors reported by survfit are usually quite different from, and usually much smaller than, the correct ones as reported by nested.km.

increase of 1 quartile in zinc) on esophageal cancer, while controlling for sex, age (as agepill, a continuous variable), smoking, drinking (both ever/never), baseline histology, and family history (yes/no). We use the same sampling model ec01*basehist as before. The output is edited for space:

```
> coxmod <- nested.coxph(coxformula =
+   "Surv(futime01,ec01==1)~sex+agepill+basehist+
     anyhist+zncent",
+   samplingmod = "ec01*basehist", data = zinc)

> summary(coxmod)
[...]
                          exp(coef) lower/upper.95
sexMale                        0.83    0.38   1.79
agepill                        1.04    0.99   1.10
basehistEsophagitis            2.97    1.41   6.26
basehistMild Dysplasia         4.88    2.19  10.88
basehistModerate Dysplasia     6.95    2.63  18.38
basehistSevere Dysplasia      11.05    3.37  36.19
basehistNOS                    3.03    0.29  30.93
basehistCIS                   34.43   10.33 114.69
anyhistFamily History          1.32    0.61   2.83
zncent                         0.73    0.57   0.93

[...]
Wald test = 97.5  on 10 df,   p=2.22e-16
```

This is the exact same coxph output, except that the R^2, overall likelihood ratio and overall score tests are not computed. The overall Wald test is correctly computed.

nested.coxph has the following restrictions

1. All variables are in the dataframe in the 'data' argument.
2. No variable in the dataframe can be named o.b.s.e.r.v.e.d. or p.i.h.a.t.
3. You must use Breslow tie-breaking.
4. No 'cluster' statements are allowed.

However, nested.coxph does allow staggered entry into the cohort, stratification of the baselize hazard via 'strata', and use of 'offset' arguments to coxph (see help for coxph for more information).

Standardized survival and attributable risk

nested.stdsurv first estimates hazard ratios exactly like nested.coxph, and then also estimates survival probabilities for each exposure level as well as Population Attributable Risk (PAR) for a given exposure level, standardizing both to the marginal confounder distribution in the cohort. For example, the standardized survival associated with exposure Q and confounder J is

$$S_{std}(t|Q) = \int S(t|J,Q)dF(J).$$

In contrast, the crude observed survival is

$$S_{crude}(t|Q) = \int S(t|J,Q)dF(J|Q).$$

The crude S is the observed survival, so the effects of confounders remain. The standardized S is estimated by using the observed J distribution as the standard, so J is independent of Q. For more on direct standardization, see Breslow and Day (1987)

To standardize, the formula for a Cox model must be split in two pieces: the argument 'exposures' denotes the part of the formula for the exposures of interest, and 'confounders' which denotes the part of the formula for the confounders. All variables in either part of the formula must be factors. In either part, do not use '*' to specify interaction, use interaction.

In the zinc example, the exposures are 'exposures="znquartiles"', a factor variable denoting which quartile of zinc each measurement is in. The confounders are 'confounders="sex+agestr+basehist+anyhist"', these are the same confounders in the hazard ratio example, except that we must categorize age as the factor agestr. 'timeofinterest' denotes the time at which survival probabilities and PAR are to be calculated at, the default is at the last event time. 'exposureofinterest' is the name of the exposure level to which the population is to be set at for computing PAR; 'exposureofinterest="Q4"' denotes that we want PAR if we could move the entire population's zinc levels into the fourth quartile of the current zinc levels. 'plot' plots the standardized survivals with 95% confidence bounds at 'timeofinterest' and returns the data used to make the plot. The output is edited for space.

```
> mod <- nested.stdsurv(outcome =
+  "Surv(futime01,ec01==1)",
+  exposures = "znquartiles",
+  confounders = "sex+agestr+basehist+anyhist",
+  samplingmod = "ec01*basehist",
+  exposureofinterest = "Q4", plot = T, main =
+  "Time to Esophageal Cancer
                          by Quartiles of Zinc",
+  data = zinc)

Std Survival for znquartiles by time 5893
      Survival  StdErr 95% CI Left 95% CI Right
Q1      0.5054 0.06936      0.3634       0.6312
Q2      0.7298 0.07768      0.5429       0.8501
Q3      0.6743 0.07402      0.5065       0.7959
Q4      0.9025 0.05262      0.7316       0.9669
Crude   0.7783 0.02283      0.7296       0.8194

Std Risk Differences vs.
                  znquartiles = Q4 by time 5893
        Risk Difference  StdErr       95% CI
Q4 - Q1          0.3972 0.09008  0.22060 0.5737
Q4 - Q2          0.1727 0.09603 -0.01557 0.3609
Q4 - Q3          0.2282 0.08940  0.05294 0.4034
```

```
Q4 - Crude        0.1242 0.05405  0.01823 0.2301

PAR if everyone had znquartiles = Q4
  Estimate StdErr 95% CI Left 95% CI Right
PAR 0.5602 0.2347     -0.2519       0.8455
```

The first table shows the survival for each quartile of zinc, standardized for all the confounders, as well as the 'crude' survival, which is the observed survival in the population (so is not standardized). The next table shows the standardized survival differences vs. the exposure of interest. The last table shows the PAR, and the CI for PAR is based on the $\log(1 - \text{PAR})$ transformation (this is often very different from, and superior to, the naive CI without transformation). `summary(mod)` yields the same hazard ratio output as if the model had been run under `nested.coxph`.

The plot is in figure 2. This plots survival curves; to plot cumulative incidence (1-survival), use '`cuminc=TRUE`'. The 95% CI bars are plotted at `timeofinterest`. All `plot` options are usable: e.g. '`main`' to title the plot.

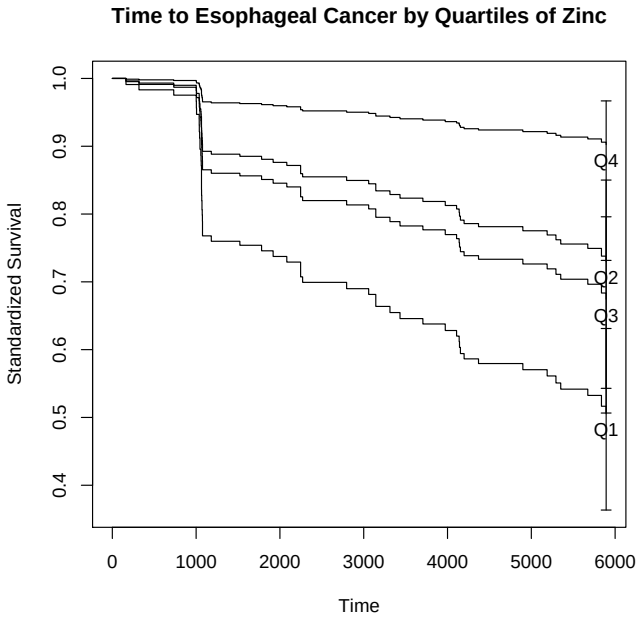

Figure 2: Survival curves for each zinc quantile, standardized for confounders

`nested.stdsurv` has some restrictions:

1. All variables are in the dataframe in the '`data`' argument.

2. No variable in the dataframe can be named `o.b.s.e.r.v.e.d.` or `p.i.h.a.t.`

3. The variables in the '`exposures`' and '`confounders`' must be factors, even if they are binary. In these formulas, never use '*' to mean interaction, use `interaction`.

4. It does not support staggered entry into the cohort.

5. It does not support the baseline hazard to be stratified. '`cluster`' and '`offset`' arguments are not supported either.

6. It only allows Breslow tie-breaking.

Including auxiliary variables

In this analysis, we used the smallest correctly-specified logistic model to predict sampling probabilities. To illustrate the use of an auxiliary variable, let's pretend we have a categorical surrogate named `znauxiliary`, a cheaply-available but non-ideal measure of zinc concentration, observed on the full cohort. The user could sample based on `znauxiliary` to try to improve efficiency. In this case, `znauxiliary` must be included as a sampling variable in the sampling model with `samplingmod="ec01*basehist*znauxiliary"`. Note that auxiliary variables must be observed on the entire cohort.

Even if sampling is not based on `znauxiliary`, it can still be included in the sampling model as above. This is because, even though `znauxiliary` was not explicitly sampled on, if `znauxiliary` has something to do with zinc, and zinc has something to do with either `ec01` or `basehist`, then one is implicitly sampling on `znauxiliary`. The simulations in (Mark and Katki (2006)) show the efficiency gain from including auxiliary variables in the sampling model. Including auxiliary variables will always reduce the standard errors of the risk estimates.

Multiple exposures

Multiple exposures (with missing values) are included in the risk regression just like any other variable. For example, if we want to estimate the esophageal cancer risk from zinc and calcium jointly, the Cox model would include `cacent` as a covariate. Cutting calcium into quartiles into the variable `caquartiles`, include it as an exposure with `nested.stdsurv` with '`exposures="znquartiles+caquartiles"`'.

Full cohort analysis

NestedCohort can be used if all covariates are observed on the full cohort. You can estimate the standardized survival and attributable risks by setting '`samplingModel="1"`', to force equal weights for all cohort members. `nested.km` will work exactly as `survfit` does. The Cox model standard errors will be those obtained from `coxph` with '`robust=TRUE`'.

Bibliography

Abnet, C. C., Lai, B., Qiao, Y.-L., Vogt, S., Luo, X.-M., Taylor, P. R., Dong, Z.-W., Mark, S. D., and Dawsey, S. M. (2005). Zinc concentration in esophageal biopsies measured by x-ray fluorescence and cancer risk. *Journal of the National Cancer Institute*, 97(4):301–306.

Breslow, N. E. and Day, N. E. (1987). *Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies.* IARC Scientific Publications, Lyon.

Mark, S. D. (2003). Nonparametric and semiparametric survival estimators,and their implementation, in two-stage (nested) cohort studies. *Proceedings of the Joint Statistical Meetings*, 2675–2691.

Mark, S. D. and Katki, H. A. (2001). Influence Function Based Variance Estimation and Missing Data Issues in Case-Cohort Studies. *Lifetime Data Analysis*, 7:331–344.

Mark, S. D. and Katki, H. A. (2006). Specifying and implementing nonparametric and semiparametric survival estimators in two-stage (sampled) cohort studies with missing case data. *Journal of the American Statistical Association*, 101(474):460–471.

Hormuzd A. Katki
Division of Cancer Epidemiology and Genetics
National Cancer Institute, NIH, DHHS, USA
katkih@mail.nih.gov

Steven D. Mark
Department of Preventive Medicine and Biometrics
University of Colorado Health Sciences Center
Steven.Mark@UCHSC.edu

segmented: An R package to Fit Regression Models with Broken-Line Relationships

by Vito M. R. Muggeo

Introduction

Segmented or *broken-line* models are regression models where the relationships between the response and one or more explanatory variables are piecewise linear, namely represented by two or more straight lines connected at unknown values: these values are usually referred as breakpoints, changepoints or even joinpoints. Hereafter we use such words indistinctly.

Broken-line relationships are common in many fields, including epidemiology, occupational medicine, toxicology, and ecology, where sometimes it is of interest to assess threshold value where the effect of the covariate changes (Ulm, 1991; Betts et al., 2007).

Formulating the model, estimation and testing

A segmented relationship between the mean response $\mu = E[Y]$ and the variable Z, for observation $i = 1, 2, \ldots, n$ is modelled by adding in the linear predictor the following terms

$$\beta_1 z_i + \beta_2 (z_i - \psi)_+ \qquad (1)$$

where $(z_i - \psi)_+ = (z_i - \psi) \times I(z_i > \psi)$ and $I(\cdot)$ is the indicator function equal to one when the statement is true. According to such parameterization, β_1 is the left slope, β_2 is the difference-in-slopes and ψ is the breakpoint. In this paper we tacitly assume a GLM with a known link function and possible additional covariates, x_i, with linear parameters δ, namely $link(\mu_i) = x_i'\delta + \beta_1 z_i + \beta_2 (z_i - \psi)_+$; however, since the discussed methods only depend on (1), we leave out from our presentation the response, the link function, and the possible linear covariates.

Breakpoints and slopes of such segmented relationship are usually of main interest, although parameters relevant to the additional covariates may be of some concern. Difficulties in estimating and testing problems are well-known in such models, see for instance Hinkley (1971). A simple and common approach to estimate the model is via grid-search type algorithms: basically, given a grid of possible candidate values of $\{\psi_k\}_{k=1,\ldots,K}$, one fits K linear models and seeks for the value corresponding to the model with the best fit. There are at least two drawbacks in using this procedure: (i) estimation might be quite cumbersome with more than one breakpoint and/or with large datasets and (ii) depending on sample size and configuration of data, estimating the model with fixed changepoint may lead the standard error of the other parameters to be too narrow, since uncertainty in the breakpoint is not taken into account.

The package **segmented** offers facilities to estimate and summarize generalized linear models with segmented relationships; virtually, no limit on the number of segmented variables and on the number of changepoint exists. **segmented** uses a method that allows the modeler to estimate simultaneously all the model parameters yielding also, at the possible convergence, the approximate full covariance matrix.

Estimation

Muggeo (2003) shows that the nonlinear term (1) has an approximate intrinsic linear representation which, to some extent, allows us to translate the problem into the standard linear framework: given an initial guess for the breakpoint, $\tilde{\psi}$ say, **segmented** attempts to estimate model (1) by fitting iteratively the linear model with linear predictor

$$\beta_1 z_i + \beta_2 (z_i - \tilde{\psi})_+ + \gamma I(z_i > \tilde{\psi})^- \qquad (2)$$

where $I(\cdot)^- = -I(\cdot)$ and γ is the parameter which may be understood as a re-parameterization of ψ and therefore accounts for the breakpoint estimation. At each iteration, a standard linear model is fitted, and the breakpoint value is updated via $\hat{\psi} = \tilde{\psi} + \hat{\gamma}/\hat{\beta}_2$; note that $\hat{\gamma}$ measures the gap, at the current estimate of ψ, between the two fitted straight lines coming from model (2). When the algorithm converges, the 'gap' should be small, i.e. $\hat{\gamma} \approx 0$, and the standard error of $\hat{\psi}$ can be obtained via the Delta method for the ratio $\frac{\hat{\gamma}}{\hat{\beta}_2}$ which reduces to $\mathrm{SE}(\hat{\gamma})/|\hat{\beta}_2|$ if $\hat{\gamma} = 0$.

The idea may be used to fit multiple segmented relationships, only by including in the linear predictor the appropriate constructed variables for the additional breakpoints to be estimated: at each step, every breakpoint estimate is updated through the relevant 'gap' and 'difference-in-slope' coefficients. Due to its computational facility, the algorithm is able to perform multiple breakpoint estimation in a very efficient way.

 ISSN 1609-3631

Testing for a breakpoint

If the breakpoint does not exist the difference-in-slopes parameter has to be zero, then a natural test for the existence of ψ is

$$H_0 : \beta_2(\psi) = 0. \quad (3)$$

Note that here we write $\beta_2(\psi)$ to stress that the parameter of interest, β_2, depends on a nuisance parameter, ψ, which vanishes under H_0. Conditions for validity of standard statistical tests (Wald, for instance) are not satisfied. More specifically, the p-value returned by classical tests is heavily underestimated, with an empirical levels about three to five times larger than the nominal levels. **segmented** employs the Davies (1987) test for performing hypothesis (3). It works as follows: given K fixed ordered values of breakpoints $\psi_1 < \psi_2 < \ldots < \psi_K$ in the range of $\mathcal{Z}$, and relevant K values of the test statistic $\{S(\psi_k)\}_{k=1,\ldots,K}$ having a standard Normal distribution for fixed ψ_k, Davies provides an upper bound given by

$$p\text{-value} \approx \Phi(-M) + V \exp\{-M^2/2\}(8\pi)^{-1/2} \quad (4)$$

where $M = max\{S(\psi_k)\}_k$ is the maximum of the K test statistics, $\Phi(\cdot)$ is the standard Normal distribution function, and $V = \sum_k(|S(\psi_k) - S(\psi_{k-1})|)$ is the total variation of $\{S(\psi_k)\}_k$. Formula (4) is an upper bound, hence the reported p-value is somewhat overestimated and the test is slightly conservative. Davies does not provide guidelines for selecting number and location of the fixed values $\{\psi_k\}_k$, however a reasonable strategy is to use the quantiles of the distribution of $\mathcal{Z}$; some simulation experiments have shown that $5 \leq K \leq 10$ usually suffices. Formula (4) refers to one-sided hypothesis test, the alternative being $H_1 : \beta_2(\psi) > 0$. The p-value for the 'lesser' alternative is obtained by using $M = min\{S(\psi_k)\}_k$, while for the two-sided case let $M = max\{|S(\psi_k)|\}_k$ and double the (4) (Davies, 1987).

The Davies test is appropriate for testing for a breakpoint, but it does not appear useful for selecting the number of the joinpoints. Following results by Tiwari et al. (2005), we suggest using the BIC for this aim.

Examples

Black dots in Figure 1 plotted on the logit scale, show the percentages of babies with Down Syndrome (DS) on births for mothers with different age groups (Davison and Hinkley, 1997, p.371). It is well-known that the risk of DS increases with the mother's age, but it is important to assess where and how such a risk changes with respect to the mother age. Presumably, at least three questions have to answered: (i) does the mother's age increase the risk of DS?; (ii) is the risk constant over the whole range of age? and (iii) if the risk is age-dependent, does a threshold value exist?

In a wider context, the problem is to estimate the broken-line model and to provide point estimates and relevant uncertainty measures of all the model parameters. The steps to be followed are straightforward with **segmented**. First, a standard GLM is estimated and a broken-line relationship is added afterwards by re-fitting the overall model. The code below uses the dataframe `down` shipped with the package.

```
> library("segmented")
> data("down")
> fit.glm<-glm(cases/births~age, weight=
+   births, family=binomial, data=down)
> fit.seg<-segmented(fit.glm, seg.Z=~age,
+   psi=25)
```

`segmented` takes the original (G)LM object (`fit.glm`) and fits a new model taking into account the piecewise linear relationship. The argument `seg.Z` is a formula (without response) which specifies the variable, possibly more than one, supposed to have a piecewise relationship, while in the `psi` argument the initial guess for the breakpoint must be supplied.

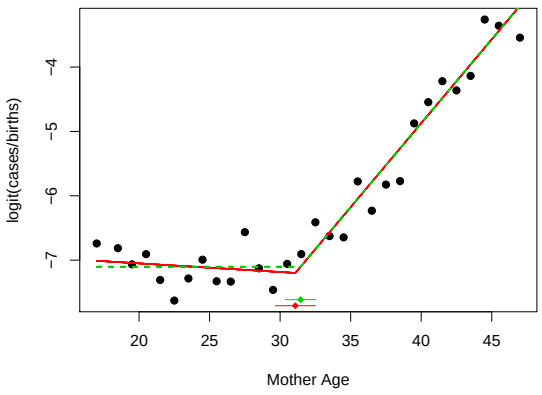

Figure 1: Scatter plot (on the logit scale) of proportion of babies with DS against mother's age and fits from models `fit.seg` and `fit.seg1`.

The estimated model can be visualized by the relevant methods `print()`, `summary()` and `print.summary()` of class `segmented`. The summary shown in Figure 2 is very similar to one of `summary.glm()`. Additional printed information include the estimated breakpoint and relevant (approximate) standard error (computed via $SE(\hat{\psi}) = SE(\hat{\gamma})/|\hat{\beta}_2|$), the t value for the 'gap' variable which should be 'small' ($|t| < 2$, say) when the algorithm converges, and the number of iterations employed to fit the model. The variable labeled with `U1.age` stands for the 'difference-in-slope parameter

```
> summary(fit.seg)

        ***Regression Model with Segmented Relationship(s)***

Call: segmented.glm(obj = fit.glm, seg.Z = ~age, psi = 25)

Estimated Break-Point(s):
    Est.  St.Err
31.0800  0.7242

t value for the gap-variable(s) V:  7.367959e-13

Meaningful coefficients of the linear terms:
               Estimate Std. Error     z value     Pr(>|z|)
(Intercept) -6.78243778 0.43140674 -15.7216777 1.074406e-55
age         -0.01341037 0.01794710  -0.7472162 4.549330e-01
U1.age       0.27422124 0.02323945  11.7998172           NA

(Dispersion parameter for binomial family taken to be 1)

Null     deviance: 625.210  on 29  degrees of freedom
Residual deviance:  43.939  on 26  degrees of freedom
AIC: 190.82

Convergence attained in 5 iterations with relative change 1.455411e-14
```

Figure 2: Output of `summary.segmented()`

of the variable age' (β_2 in equation (1)) and the estimate of the gap parameter γ is omitted since it is just a trick to estimate ψ. Note, however, that the model degrees of freedom are correctly computed and displayed.

Also notice that the p-value relevant to `U1.age` is not reported, and `NA` is printed. The reason is that, as discussed previously, standard asymptotics do not apply. In order to test for a significant difference-in-slope, the Davies' test can be used. The use of `davies.test()` is straightforward and requires to specify the regression model (`lm` or `glm`), the 'segmented' variable whose a broken-line relationship is being tested, and the number of the evaluation points,

```
> davies.test(fit.glm,"age",k=5)

        Davies' test for a change in the slope

data:  Model =  binomial , link = logit
formula = cases/births ~ age
segmented variable = age
'Best' at = 32, n.points = 5, p-value < 2.2e-16
alternative hypothesis: two.sided
```

Currently `davies.test()` only uses the Wald statistic, i.e. $S(\psi_k) = \hat{\beta}_2/\mathrm{SE}(\hat{\beta}_2)$ for each fixed ψ_k, although alternative statistics could be used.

If the breakpoint exists, the limiting distribution of $\hat{\beta}_2$ is gaussian, therefore estimates (and standard errors) of the slopes can be easily computed via the function `slope()` whose argument `conf.level` specifies the confidence level (defaults to `conf.level=0.95`),

```
> slope(fit.seg)
$age
          Est. St.Err. t value CI(95%).l CI(95%).u
slope1 -0.01341 0.01795 -0.7472  -0.04859   0.02177
slope2  0.26080 0.01476 17.6700   0.23190   0.28970
```

Davison and Hinkley (1997) discuss that it might be of some interest to test for a null left slope, and at this aim they use isotonic regression. On the other hand, the piecewise parameterization allows to face this question in a straightforward way since only a test for $H_0 : \beta_1 = 0$ has be performed; for instance, a Wald test is available directly from the summary (see Figure 2, $t = -0.747$). Under a null-left-slope constraint, a segmented model may be fitted by omitting from the 'initial' model the segmented variable, namely

```
> fit.glm<-update(fit.glm,.~.-age)
> fit.seg1<-update(fit.seg)
```

While the fit is substantially unchanged, the (approximate) standard error of the breakpoint is noticeably reduced (compare the output in Figure 2)

```
> fit.seg1$psi
         Initial     Est.    St.Err
psi1.age      25 31.45333 0.5536572
```

Instead, as firstly observed in Hinkley (1971) and shown by some simulations, the breakpoint estimator coming from a null left slope model is more efficient as compared to the one coming from a nonnull

left slope fit. Fitted values for both segmented models are displayed in Figure 1 where broken-lines and bars for the breakpoint estimates have been added via the relevant methods `plot()` and `lines()` detailed at the end of this section.

We continue our illustration of the **segmented** package by running a further example using the `plant` dataset in the package. This example may be instructive to describe how to fit multiple segmented relationships with also a zero constraint on the right slope. Data refer to variables, `y`, `time` and `group` which represent measurements of a plant organ over time for three attributes (levels of the factor `group`). The data have been kindly provided by Dr Zongjian Yang at School of Land, Crop and Food Sciences, The University of Queensland, Brisbane, Australia. Biological reasoning and empirical evidence as emphasized in Figure 3, indicate that non-parallel segmented relationships with multiple breakpoints may allow a well-grounded and reasonable fit. Multiple breakpoints are easily accounted in equation (1) by including additional terms $\beta_3(z_i - \psi_2)_+ + \ldots$ `segmented` allows a such extension in a straightforward manner by supplying multiple starting points in the `psi` argument.

To fit such a broken-line model within **segmented**, we first need to build the three different explanatory variables, products of the covariate `time` by the dummies of `group` [1],

```
> data("plant")
> attach(plant)
> X<-model.matrix(~0+group)*time
> time.KV<-X[,1]
> time.KW<-X[,2]
> time.WC<-X[,3]
```

Then we call `segmented` on a `lm` fit, by specifying multiple segmented variables in `seg.Z` and using a list to supply the starting values for the breakpoints in `psi`. We assume two breakpoints in each series,

```
> olm<-lm(y~0+group+ time.KV + time.KW + time.WC)
> os<-segmented(olm, seg.Z= ~ time.KV + time.KW
+   + time.WC, psi=list(time.KV=c(300,450),
+   time.KW=c(450,600), time.WC=c(300,450)))
Warning message:
max number of iterations attained
```

Some points are probably worth mentioning here. First, the starting linear model `olm` could be fitted via the more intuitive call `lm(y~group*time)`: even if `segmented()` would have worked providing the same results, a possible use of `slope()` would have not been allowed. Second, since there are multiple segmented variables, the starting values - obtained by visual inspection of the scatter-plots - have to supplied via a named list whose names have to match with the variables in `seg.Z`. Last but not least, the printed message suggests to re-fit the model because convergence is suspected. Therefore it could be helpful to trace out the algorithm and/or to increase the maximum number of the iterations,

```
> os<-update(os, control=seg.control(it.max=30,
+     display=TRUE))
0   1.433  (No breakpoint(s))
1   0.108
2   0.109
3   0.108
4   0.109
5   0.108
. . . .
29  0.108
30  0.109
Warning message:
max number of iterations attained
```

The optimized objective function (residual sum of squares in this case) alternates among two values and 'does not converge', in that differences never reach the (default) tolerance value of 0.0001; the function `draw.history()` may be used to visualize the values of breakpoints throughout the iterations. Moreover, increasing the number of maximum iterations, typically does not modify the result. This is not necessarily a problem. One could change the tolerance by setting `toll=0.001`, say, or better, stop the algorithm at the iteration with the best value. Also, one could stabilize the algorithm by shrinking the increments in breakpoint updates through a factor $h < 1$, say; this is attained via the argument `h` in the auxiliary function `seg.control()`,

```
> os<-update(os, control=seg.control(h=.3))
```

However, when convergence is not straightforward, the fitted model has to be inspected with particular care: if a breakpoint is understood to exist, the corresponding difference-in-slope estimate (and its t value) has to be large and furthermore the 'gap' coefficient (and its t value) has to be small (see the `summary(..)$gap`). If at the estimated breakpoint the coefficient of the gap variable is large (greater than two, say) a broken-line parameterization is somewhat questionable. Finally, a test for the existence of the breakpoint and/or comparing the BIC values would be very helpful in these circumstances.

Green diamonds in Figure 3 and output from `slope()` (not shown) show that the last slope for group "KW" may be set to zero. While a left slope is allowed by fitting only $(z - \psi)_+$ (i.e. by omitting the main variable z in the initial linear model as in the previous example), similarly a null right slope might be allowed by including only $(z - \psi)_-$. **segmented** does not handle such terms explicitly, however by noting that $(z - \psi)_- = -(-z + \psi)_+$, we can proceed as follows

[1] Of course, a corner-point parameterization (i.e. 'treatment' contrasts) is required to define the dummies relevant to the grouping variable; this is the default in R.

```
> neg.time.KW<- -time.KW
> olm1<-lm(y~0+group+time.KV+time.WC)
> os1<-segmented(olm1, seg.Z=~ time.KV + time.WC+
+  neg.time.KW, psi=list(time.KV=c(300,450),
+  neg.time.KW=c(-600,-450), time.WC=c(300,450)))
```

The 'minus' of the explanatory variable in group "KW" requires that the corresponding starting guess has to be supplied with reversed sign and, as consequence, the signs of estimates for the corresponding group will be reversed. The method `segmented` for `confint()` may be used to display (large sample) interval estimates for the breakpoints; confidence intervals are computed using $\hat{\psi} \mp z_{\alpha/2}\mathrm{SE}(\hat{\psi})$ where $\mathrm{SE}(\hat{\psi})$ comes from the Delta method for the ratio $\frac{\hat{\gamma}}{\hat{\beta}_2}$ and $z_{\alpha/2}$ is the quantile of the standard Normal. Optional arguments are `parm` to specify the segmented variable of interest (default to all variables) and `rev.sgn` to change the sign of output before printing (this is useful when the sign of the segmented variable has been changed to constrain the last slope as in example at hand).

```
> confint(os1,rev.sgn=c(FALSE,FALSE,TRUE))
$time.KV
               Est. CI(95%).l CI(95%).u
psi1.time.KV 299.9     256.9     342.8
psi2.time.KV 441.9     402.0     481.8

$time.WC
               Est. CI(95%).l CI(95%).u
psi1.time.WC 306.0     284.2     327.8
psi2.time.WC 460.1     385.5     534.7

$neg.time.KW
                  Est. CI(95%).l CI(95%).u
psi1.neg.time.KW 445.4     398.5     492.3
psi2.neg.time.KW 609.9     549.7     670.0
```

The slope estimates may be obtained using `slope()`; again, `parm` and `rev.sgn` may be specified when requested,

```
> slope(os1, parm="neg.time.KW", rev.sgn=TRUE)
$neg.time.KW
            Est.   St.Err. t value CI(95%).l CI(95%).u
slope1 0.0022640 8.515e-05  26.580 0.0020970  0.002431
slope2 0.0008398 2.871e-04   2.925 0.0002771  0.001403
slope3 0.0000000        NA      NA        NA        NA
```

Notice that in light of the constrained right slope, standard errors, t-values, and confidence limits are not computed.

Figure 3 emphasizes the constrained fit which has been added to the current device via the relevant `plot()` method. More specifically, `plot()` allows to draw on the current or new device (depending on the logical value `TRUE/FALSE` of `add`) the fitted piecewise relationship for the variable `term`. To get sensible plots with fitted values to be superimposed to the observed points, the arguments `const` and `rev.sgn` have to be set carefully. The role of `rev.sgn` is intuitive and has been discussed above while `const` indicates a constant to be added to the fitted values before plotting,

```
> plot(os1, term="neg.time.KW", add=TRUE, col=3,
+   const=coef(os1)["groupRKW"], rev.sgn=TRUE)
```

`const` defaults to the model intercept, and for relationships by group the group-specific intercept is appropriate, as in the "KW" group example above. However when a 'minus' variable has been considered, simple algebra on the regression equation show that the correct constant for the other groups is given by the current estimate minus a linear combination of difference-in-slope parameters and relevant breakpoints. For the "KV" group we add the fitted lines after computing the 'adjusted' constant,

```
> const.KV<-coef(os1)["groupRKV"]-
+   coef(os1)["U1.neg.time.KW"]*
+   os1$psi["psi1.neg.time.KW","Est."]-
+   coef(os1)["U2.neg.time.KW"]*
+   os1$psi["psi2.neg.time.KW","Est."]
> plot(os1, "time.KV", add=TRUE, col=2, const=const.KV)
```

and similarly for group "WC".

Finally the estimated join points with relevant confidence intervals are added to the current device via the `lines.segmented()` method,

```
> lines(os1,term="neg.time.KW",col=3,rev.sgn=TRUE)
> lines(os1,term="time.KV",col=2,k=20)
> lines(os1,term="time.WC",col=4,k=10)
```

where `term` selects the segmented variable, `rev.sgn` says if the sign of the breakpoint values (point estimate and confidence limits) have to be reversed, `k` regulates the vertical position of the bars, and the remaining arguments refer to options of the drawn segments.

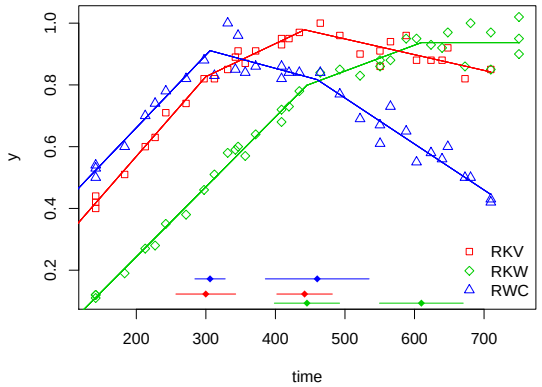

Figure 3: The `plant` dataset: data and constrained fit (model `os1`).

Conclusions

We illustrated the key-ideas of broken-line regression and how such a class of models may be fitted

in R through the package **segmented**. Although alternative approaches could be undertaken to model nonlinear relationships, for instance via splines, the main appealing of segmented models lies on interpretability of the parameters. Sometimes a piecewise parameterization may provide a reasonable approximation to the shape of the underlying relationship, and threshold and slopes may be very informative and meaningful.

However it is well known that the likelihood in segmented models may not be concave, hence there is no guarantee the algorithm finds the global maximum; moreover it should be recognized that the method works by approximating the 'true' model (1) by (2), which could make the estimation problematic. A possible and useful strategy - quite common in the nonlinear optimization field - is to run the algorithm starting with different initial guesses for the breakpoint in order to assess possible differences. This is quite practicable due to computational efficiency of the algorithm. However, the more the clear-cut the relationship, the less important the starting values become.

The package is not concerned with estimation of the number of the breakpoints. Although the BIC has been suggested, in general nonstatistical issues related to the understanding of the mechanism of the phenomenon in study could help to discriminate among several competing models with a different number of joinpoints.

Currently, only methods for LM and GLM objects are implemented; however, due to the ease of the algorithm which only depends on the linear predictor, methods for other models (Cox regression, say) could be written straightforwardly following the skeleton of `segmented.lm` or `segmented.glm`.

Finally, for the sake of novices in breakpoint estimation, it is probably worth mentioning the difference existing with the other R package dealing with breakpoints. The **strucchange** package by Zeileis et al. (2002) substantially is concerned with regression models having a different set of parameters for each 'interval' of the segmented variable, typically the time; **strucchange** performs breakpoint estimation via a dynamic grid search algorithm and allows for testing for parameter instability. Such 'structural breaks models', mainly employed in economics and econometrics, are somewhat different from the broken-line models discussed in this paper, since they do not require the fitted lines to join at the estimated breakpoints.

Acknowledgements

This work was partially supported by grant 'Fondi di Ateneo (ex 60%) 2004 prot. ORPA044431: 'Verifica di ipotesi in modelli complessi e non-standard' ('Hypothesis testing in complex and nonstandard models'). The author would like to thank the anonymous referee for useful suggestions which improved the paper and the interface of the package itself.

Bibliography

M. Betts, G. Forbes, and A. Diamond. Thresholds in songbird occurrence in relation to landscape structure. *Conservation Biology*, 21:1046–1058, 2007.

R. B. Davies. Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika*, 74:33–43, 1987.

A. Davison and D. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997.

D. Hinkley. Inference in two-phase regression. *Journal of American Statistical Association*, pages 736–743, 1971.

V. Muggeo. Estimating regression models with unknown break-points. *Statistics in Medicine*, 22: 3055–3071, 2003.

R. Tiwari, K. A. Cronin, W. Davis, E. Feuer, B. Yu, and S. Chib. Bayesian model selection for join point regression with application to age-adjusted cancer rates. *Applied Statistics*, 54:919–939, 2005.

K. Ulm. A statistical methods for assessing a threshold in epidemiological studies. *Statistics in Medicine*, 21:341–349, 1991.

A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. `strucchange`: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38, 2002.

Vito M. R. Muggeo
Dipartimento Scienze Statistiche e Matematiche 'Vianelli'
Università di Palermo, Italy
`vmuggeo@dssm.unipa.it`

Bayesian Estimation for Parsimonious Threshold Autoregressive Models in R

by Cathy W.S. Chen, Edward M.H. Lin, F.C. Liu, and Richard Gerlach

Introduction

The threshold autoregressive (TAR) model proposed by Tong (1978, 1983) and Tong and Lim (1980) is a popular nonlinear time series model that has been widely applied in many areas including ecology, medical research, economics, finance and others (Brockwell, 2007). Some interesting statistical and physical properties of TAR include asymmetry, limit cycles, amplitude dependent frequencies and jump phenomena, all of which linear models are unable to capture. The standard two-regime threshold autoregressive (TAR) model is considered in this paper. Given the autoregressive (AR) orders p_1 and p_2, the two-regime TAR(2:p_1;p_2) model is specified as:

$$y_t = \begin{cases} \phi_0^{(1)} + \sum_{i=1}^{p_1} \phi_{k_i}^{(1)} y_{t-k_i} + a_t^{(1)} \text{ if } z_{t-d} \leq r, \\ \phi_0^{(2)} + \sum_{i=1}^{p_2} \phi_{l_i}^{(2)} y_{t-l_i} + a_t^{(2)} \text{ if } z_{t-d} > r, \end{cases}$$

where $\{k_i, i = 1, \ldots, p_1\}$ and $\{l_i, i = 1, \ldots, p_2\}$ are subsets of $\{1, \ldots, p\}$,

where r is the threshold parameter driving the regime-switching behavior; where p is a reasonable maximum lag [1]; z_t is the threshold variable; d is the threshold lag of the model; $a_t^{(1)}$ and $a_t^{(2)}$ are two independent Gaussian white noise processes with mean zero and variance σ_j^2, $j = 1, 2$. It is common to choose the threshold variable **z** as a lagged value of the time series itself, that is $z_t = y_t$. In this case the resulting model is called Self-Exciting (SETAR). In general, **z** could be any exogenous or endogenous variable (Chen 1998). The TAR model consists of a piecewise linear AR model in each regime, defined by the threshold variable z_{t-d} and associated threshold value r. Note that the parameter p is not an input to the R program, but instead should be considered by the user as the largest possible lag the model could accommodate, e.g. in light of the sample size n. i.e. $p << n$ is usually enforced for AR models.

Frequentist parameter estimation of the TAR model is usually implemented in two stages; see for example Tong and Lim (1980), Tong (1990) and Tsay (1989, 1998). For fixed and subjectively chosen values of d (usually 1) and r (usually 0), all other model parameters are estimated first. Then, conditional on these parameter estimates, d and r can be estimated by: minimizing the AIC, minimizing a nonlinearity test statistic and/or using scatter plots (Tsay, 1989); or by minimizing a conditional least squares formula (Tsay, 1998).

Bayesian methods allow simultaneous inference on all model parameters, in this case allowing uncertainty about the threshold lag d and threshold parameter r to be properly incorporated into estimation and inference. Such uncertainty is not accounted for in the standard two-stage methods. However, in the nonlinear TAR setting, neither the marginal or joint posterior distributions for the parameters can be easily analytically obtained: these usually involve high dimensional integrations and/or non-tractable forms. However, the joint posterior distribution can be evaluated up to a constant, and thus numerical integration techniques can be used to estimate the marginal distributions required. The most successful of these, for TAR models, are Markov chain Monte Carlo (MCMC) methods, specifically those based on the Gibbs sampler. Chen and Lee (1995) proposed such a method, incorporating the Metropolis-Hastings (MH) algorithm (Metropolis *et al.*, 1953; Hastings, 1970), for inference in TAR models. Utilizing this MH within Gibbs algorithm, the marginal and posterior distributions can be estimated by iterative sampling. To the best of our knowledge, this is the first time a Bayesian approach for TAR models has been offered in a statistical package.

We propose an R package BAYSTAR that provides functionality for parameter estimation and inference for two-regime TAR models, as well as allowing the monitoring of MCMC convergence by returning all MCMC iterates as output. The main features of the package BAYSTAR are: applying the Bayesian inferential methods to simulated or real data sets; online monitoring of the acceptance rate and tuning parameters of the MH algorithm, to aid the convergence of the Markov chain; returning all MCMC iterates for user manipulation, clearly reporting the relevant MCMC summary statistics and constructing trace plots and auto-correlograms as diagnostic tools to assess convergence of MCMC chains. This allows us to statistically estimate all unknown model parameters simultaneously, including capturing uncertainty about threshold value and delay lag; not accounted for in standard methods that condition upon a particular threshold value and delay lag, see e.g. the SETAR function which is available in the "tsDyn" package at CRAN. We also allow the user to define a parsimonious separate AR order specification in each regime. Using our code it is possible to set some AR parameters in either or both regimes to be

[1] We have tried up to p=50 successfully.

zero. That is, we could set $p_1 = 3$ and subsequently estimate any three parameters of our convenience.

Prior settings

Bayesian inference requires us to specify a prior distribution for the unknown parameters. The parameters of the TAR(2:p_1;p_2) model are Θ_1, Θ_2, σ_1^2, σ_2^2, r and d, where Θ_1=($\phi_0^{(1)}$, $\phi_1^{(1)}$, …, $\phi_{p_1}^{(1)}$)′ and Θ_2=($\phi_0^{(2)}$, $\phi_1^{(2)}$, …, $\phi_{p_2}^{(2)}$)′. We take fairly standard choices: Θ_1, Θ_2 as independent $N(\Theta_{0i}, V_i^{-1})$, $i = 1, 2$, and employ conjugate priors for σ_1^2 and σ_2^2,

$$\sigma_i^2 \sim IG\left(\nu_i/2, \nu_i\lambda_i/2\right), \quad i = 1, 2,$$

where IG stands for the inverse Gamma distribution. In threshold modeling, it is important to set a minimum sample size in each regime to generate meaningful inference. The prior for the threshold parameter r, follows a uniform distribution on a range (l, u), where l and u are set as relevant percentiles of the observed threshold variable. This prior could be considered to correspond to an empirical Bayes approach, rather than a fully Bayesian one. Finally, the delay d has a discrete uniform prior over the integers: 1,2,…, d_0, where d_0 is a user-set maximum delay. We assume the hyper-parameters, $(\Theta_{0i}, V_i, \nu_i, \lambda_i, a, b, d_0)$ are known and can be specified by the user in our `R` code.

The MCMC sampling scheme successively generates iterates from the posterior distributions for groups of parameters, conditional on the sample data and the remaining parameters. Multiplying the likelihood and the priors, using Bayes' rule, leads to these conditional posteriors. For details, readers are referred to Chen and Lee (1995). Only the posterior distribution of r is not a standard distributional form, thus requiring us to use the MH method to achieve the desired sample for r. The standard Gaussian proposal random walk MH algorithm is used. To yield good convergence properties for this algorithm, the choice of step size, controlling the proposal variance, is important. A suitable value of the step size, with good convergence properties, can be achieved by tuning to achieve an acceptance rate between 25% to 50%, as suggested by Gelman, Roberts and Gilks (1996). This tuning occurs only in the burn-in period.

Exemplary applications

Simulated data

We now illustrate an example with simulated data. The data is generated from a two-regime SETAR(2 : 2; 2) model specified as:

$$y_t = \begin{cases} 0.1 - 0.4y_{t-1} + 0.3y_{t-2} + a_t^{(1)} \text{ if } y_{t-1} \leq 0.4, \\ 0.2 + 0.3y_{t-1} + 0.3y_{t-2} + a_t^{(2)} \text{ if } y_{t-1} > 0.4, \end{cases}$$

where $a_t^{(1)} \sim N(0, 0.8)$ and $a_t^{(2)} \sim N(0, 0.5)$.

Users can import data from an external file, or use their own simulated data, and directly estimate model parameters via the proposed functions. To implement the MCMC sampling, the scheme was run for $N = 10,000$ iterations (the total MCMC sample) and the first $M = 2,000$ iterations (the burn-in sample) were discarded.

```
+ nIterations<- 10000
+ nBurnin<- 2000
```

The hyper-parameters are set as $\Theta_{0i} = \mathbf{0}$, V_i = diag(0.1, …, 0.1), ν_i = 3 and $\lambda_i = \widetilde{\sigma}^2/3$ for i = 1,2, where $\widetilde{\sigma}^2$ is the residual mean squared error of fitting an AR(p_1) model to the data. The motivation to choose the hyper-parameters of ν_i and λ_i is that the expectation of σ_i^2 is equal to $\widetilde{\sigma}^2$. The maximum delay lag is set to d_0 = 3. We choose $a = Q_1$ and $b = Q_3$: the 1st and 3rd quartiles of the data respectively, for the prior on r.

```
+ mu0<- matrix(0, nrow=p1+1, ncol=1)
+ v0<- diag(0.1, p1+1)
+ ar.mse<- ar(yt,aic=FALSE, order.max=p1)
+ v<- 3; lambda<- ar.mse$var.pred/3
```

The MCMC sampling steps sequentially draw samples of parameters by using the functions `TAR.coeff()`, `TAR.sigma()`, `TAR.lagd()` and `TAR.thres()`, iteratively. `TAR.coeff()` returns the updated values of Θ_1 and Θ_2 from a multivariate normal distribution for each regime. σ_1^2 and σ_2^2 are sampled separately using the function `TAR.sigma()` from inverse gamma distributions. `TAR.lagd()` and `TAR.thres()` are used to sample d, from a multinomial distribution, and r, by using the MH algorithm, respectively. The required log-likelihood function is computed by the function `TAR.lik()`. When drawing r, we monitor the acceptance rate of the MH algorithm so as to maximize the chance of achieving a stationary and convergent sample. The `BAYSTAR` package provides output after every 1,000 MCMC iterations, for monitoring the estimation, and the acceptance rate, of r. If the acceptance rate falls outside 25% to 50%, the step size of the MH algorithm is automatically adjusted during burn-in iterations, without re-running the whole program. Enlarging the step size should reduce the acceptance rate while diminishing the step size should increase this rate.

A summary of the MCMC output can be obtained via the function `TAR.summary()`. `TAR.summary()` returns the posterior mean, median, standard deviation and the lower and upper bound of the 95%

	true	mean	median	s.d.	lower	upper
phi0^1	0.1000	0.0873	0.0880	0.0395	0.0096	0.1641
phi1^1	-0.4000	-0.3426	-0.3423	0.0589	-0.4566	-0.2294
phi2^1	0.3000	0.2865	0.2863	0.0389	0.2098	0.3639
phi0^2	0.2000	0.2223	0.2222	0.0533	0.1187	0.3285
phi1^2	0.3000	0.2831	0.2836	0.0407	0.2040	0.3622
phi2^2	0.3000	0.3244	0.3245	0.0234	0.2780	0.3701
sigma1	0.8000	0.7789	0.7773	0.0385	0.7079	0.8587
simga2	0.5000	0.5563	0.5555	0.0231	0.5132	0.6029
r	0.4000	0.4161	0.4097	0.0222	0.3968	0.4791
diff.phi0	-0.1000	-0.1350	-0.1354	0.0654	-0.2631	-0.0039
diff.phi1	-0.7000	-0.6257	-0.6258	0.0726	-0.7657	-0.4841
diff.phi2	0.0000	-0.0379	-0.0381	0.0455	-0.1256	0.0521
mean1	0.0909	0.0834	0.0829	0.0390	0.0088	0.1622
mean2	0.5000	0.5598	0.5669	0.0888	0.3673	0.7161

Lag choice :

	1	2	3
Freq	10000	0	0

Figure 1: The summary output for all parameters is printed as a table.

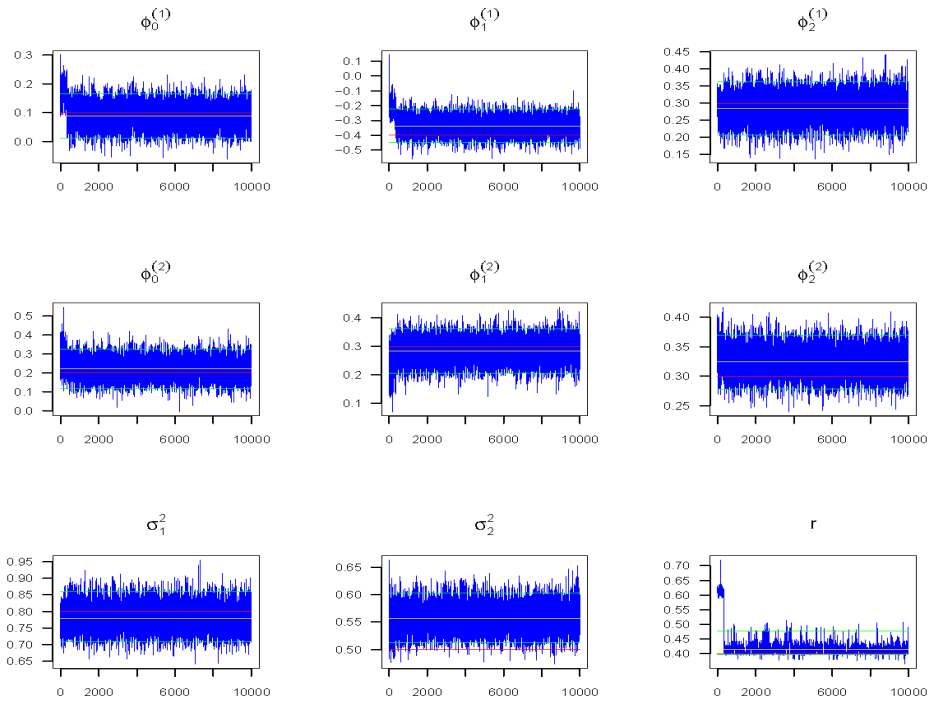

Figure 2: The trace plots of all MCMC iterations for all parameters.

Bayes posterior interval for all parameters, all obtained from the sampling period only, after burn-in. Output is also displayed for the differences in the mean coefficients and the unconditional mean in each regime. The summary statistics are printed as in Figure 1.

To assess MCMC convergence to stationarity, we monitor trace plots and autocorrelation plots of the MCMC iterates for all parameters. Trace plots are obtained via `ts.plot()` for all MCMC iterations, as shown in Figure 2. The red horizontal line is the true value of the parameter, the yellow line represents the posterior mean and the green lines are the lower and upper bounds of the 95% Bayes credible interval. The density plots, via the function `density()`, are provided for each parameter and the differences in the mean coefficients as shown in Figure 3.

An example of a simulation study is now illustrated. For 100 simulated data sets, the code saves the posterior mean, median, standard deviation and the lower and upper bound of each 95% Bayesian interval for each parameter. The means, over the replicated data sets, of these quantities are reported as a table in Figure 4. For counting the frequencies of each estimated delay lag, we provide the frequency table of d by the function `table()`, as shown in the bottom of Figure 4. The average posterior probabilities that $d = 1$ are all very close to 1; the posterior mode of d very accurately estimates the true delay parameter in this case.

US unemployment rate data

For empirical illustration, we consider the monthly U.S. civilian unemployment rate from January 1948 to March 2004. The data set, which consists of 675 monthly observations, is shown in Figure 5. The data is available in Tsay (2005). We take the first difference of the unemployment rates in order to achieve mean stationarity. A partial autocorrelation plot (PACF) of the change of unemployment rate is given in Figure 5. For illustration, we use the same model orders as in Tsay (2005), except for the addition of a 10th lag in regime one. We obtain the fitted SETAR model:

$$y_t = \begin{cases} 0.187y_{t-2} + 0.143y_{t-3} + 0.127y_{t-4} \\ -0.106y_{t-10} - 0.087y_{t-12} + a_t^{(1)} \text{ if } y_{t-3} \leq 0.05, \\ 0.312y_{t-2} + 0.223y_{t-3} - 0.234y_{t-12} \\ +a_t^{(2)} \text{ if } y_{t-3} > 0.05, \end{cases}$$

The results are shown in Figure 6. Trace plots and autocorrelograms for after burn-in MCMC iterations are given in Figures 7 and 8. Clearly, MCMC convergence is almost immediate. The parameter estimates are quite reasonable, being similar to the results of Tsay (2005), except the threshold lag, which is set as $d = 1$ by Tsay. Instead, our results suggest that nonlinearity in the differences in the unemployment rate, responds around a positive 0.05 change in the unemployment rate, is at a lag of $d = 3$ months, for this data. This is quite reasonable. The estimated AR coefficients differ between the two regimes, indicating the dynamics of the US unemployment rate are based on the previous quarter's change in rate. It is also clear that the regime variances are significantly different to each other, which can be confirmed by finding a 95% credible interval from the MCMC iterates of the differences between these parameters.

Summary

`BAYSTAR` provides Bayesian MCMC methods for iterative sampling to provide parameter estimates and inference for the two-regime TAR model. Parsimonious AR specifications between regimes can also be easily employed. A convenient user interface for importing data from a file or specifying true parameter values for simulated data is easy to apply for analysis. Parameter inferences are summarized to an easily readable format. Simultaneously, the checking of convergence can be done by monitoring the MCMC trace plots and autocorrelograms. Simulations illustrated the good performance of the sampling scheme, while a real example illustrated nonlinearity present in the US unemployment rate. In the future we will extend `BAYSTAR` to more flexible models, such as threshold moving-average (TMA) models and threshold autoregressive moving-average (TARMA) models, which are also frequently used in time series modeling. In addition model and order selection is an important issue for these models. It is interesting to examine the method of the stochastic search variable selection (SSVS) in the `R` package with `BAYSTAR` for model order selection in these types of models, e.g. So and Chen (2003).

Acknowledgments

Cathy Chen thanks Professor Kerrie Mengersen for her invitation to appear as keynote speaker and to present this work at the Spring Bayes 2007 workshop. Cathy Chen is supported by the grant: 96-2118-M-002-MY3 from the National Science Council (NSC) of Taiwan and grant 06G27022 from Feng Chia University. The authors would like to thank the editors and anonymous referee for reading and assessing the paper, and especially to thank the referee who made such detailed and careful comments that significantly improved the paper.

Bibliography

P. Brockwell. Beyond Linear Time Series. *Statistica Sinica*, 17:3-7, 2007.

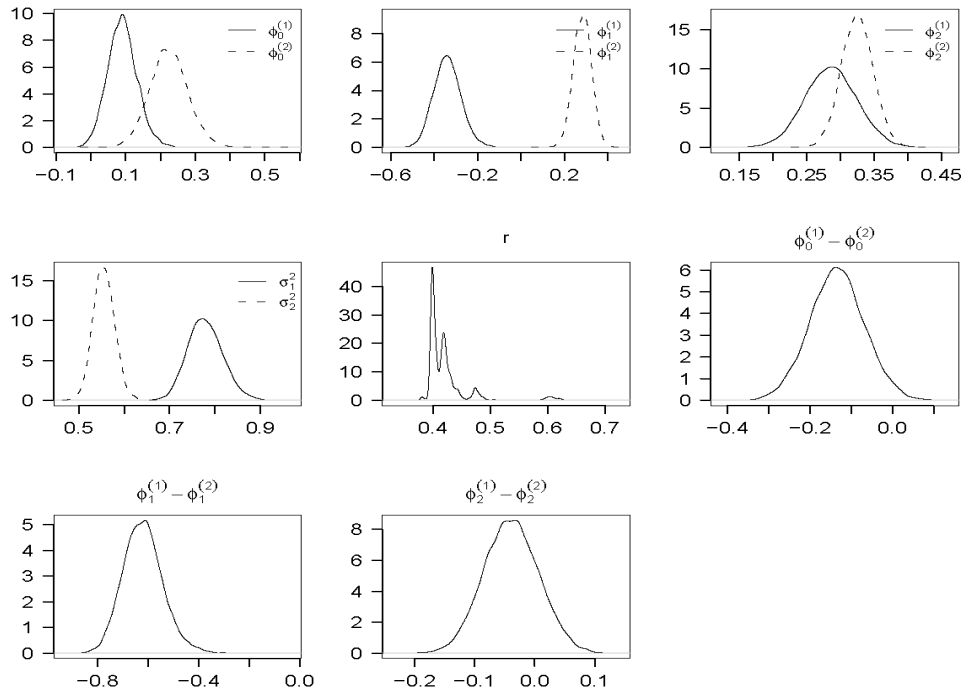

Figure 3: Posterior densities of all parameters and the differences of mean coefficients.

```
              true    mean  median   s.d.   lower   upper
phi0^1      0.1000  0.0917  0.0918 0.0401  0.0129  0.1701
phi1^1     -0.4000 -0.4058 -0.4056 0.0616 -0.5268 -0.2852
phi2^1      0.3000  0.3000  0.3000 0.0417  0.2184  0.3817
phi0^2      0.2000  0.2082  0.2082 0.0509  0.1088  0.3082
phi1^2      0.3000  0.2940  0.2940 0.0387  0.2181  0.3697
phi2^2      0.3000  0.2961  0.2961 0.0226  0.2517  0.3404
sigma1      0.8000  0.7979  0.7966 0.0397  0.7239  0.8796
simga2      0.5000  0.5038  0.5033 0.0209  0.4645  0.5464
r           0.4000  0.3944  0.3948 0.0157  0.3657  0.4247
diff.phi0 -0.1000 -0.1165 -0.1166 0.0644 -0.2425  0.0099
diff.phi1 -0.7000 -0.6997 -0.6995 0.0731 -0.8431 -0.5568
diff.phi2  0.0000  0.0039  0.0040 0.0475 -0.0890  0.0972
mean1      0.0909  0.0841  0.0836 0.0379  0.0116  0.1601
mean2      0.5000  0.4958  0.5024 0.0849  0.3109  0.6434
> table(lag.yt)
lag.yt
  1
100
```

Figure 4: The summary output for all parameters from 100 replications is printed as a table.

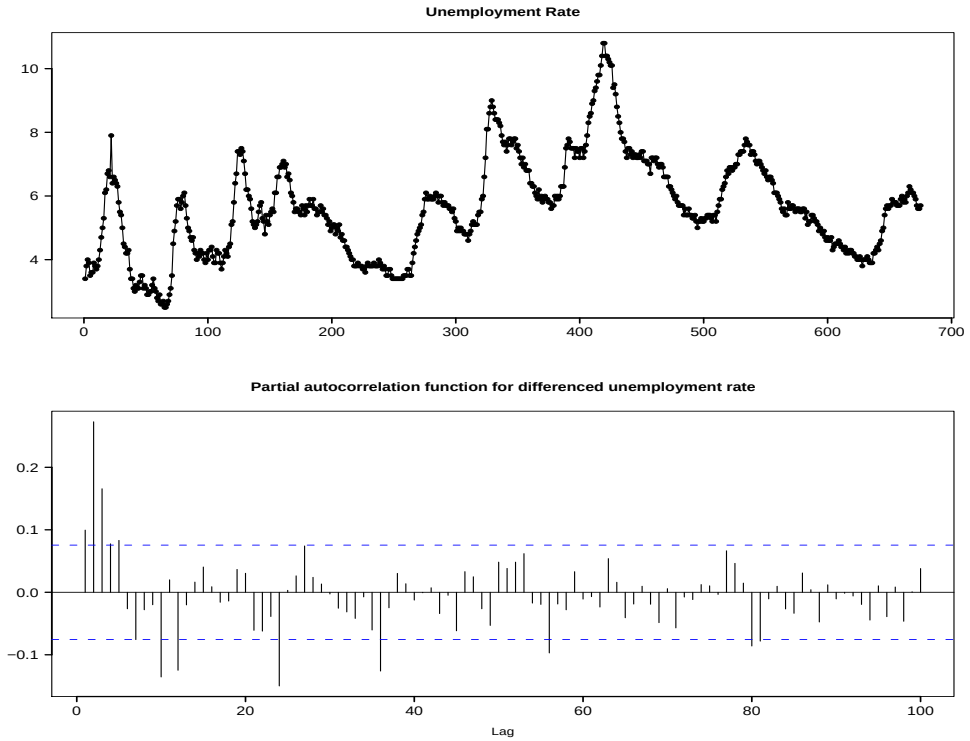

Figure 5: Time series plots of the PACF of the changed unemployment rate.

```
           mean  median   s.d.   lower   upper
phi1.2   0.1874  0.1877 0.0446  0.0993  0.2751
phi1.3   0.1431  0.1435 0.0457  0.0526  0.2338
phi1.4   0.1270  0.1273 0.0447  0.0394  0.2157
phi1.10 -0.1060 -0.1058 0.0400 -0.1855 -0.0275
phi1.12 -0.0875 -0.0880 0.0398 -0.1637 -0.0082
phi2.2   0.3121  0.3124 0.0613  0.1932  0.4349
phi2.3   0.2233  0.2233 0.0594  0.1077  0.3387
phi2.12 -0.2340 -0.2341 0.0766 -0.3837 -0.0839
sigma1   0.0299  0.0298 0.0021  0.0261  0.0342
simga2   0.0588  0.0585 0.0054  0.0492  0.0702
r        0.0503  0.0506 0.0290  0.0027  0.0978
Lag choice :
      1 2    3
Freq 15 0 9985
------------
The highest posterior prob. of lag at :  3
```

Figure 6: The summary output for all parameters of the U.S. unemployment rate is printed as a table.

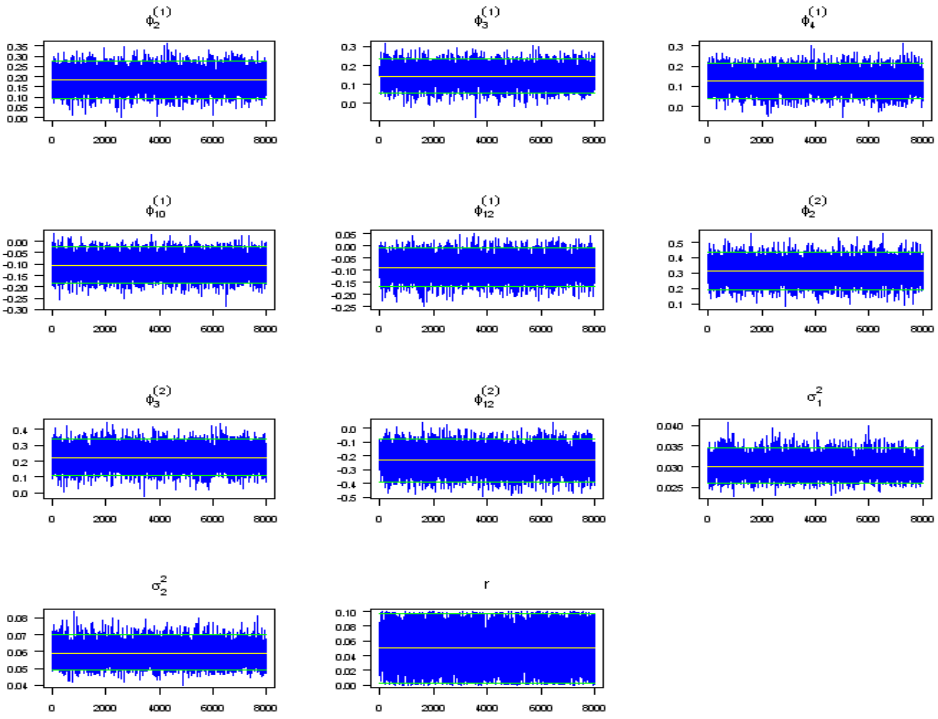

Figure 7: The trace plots of after burn-in MCMC iterations for all parameters.

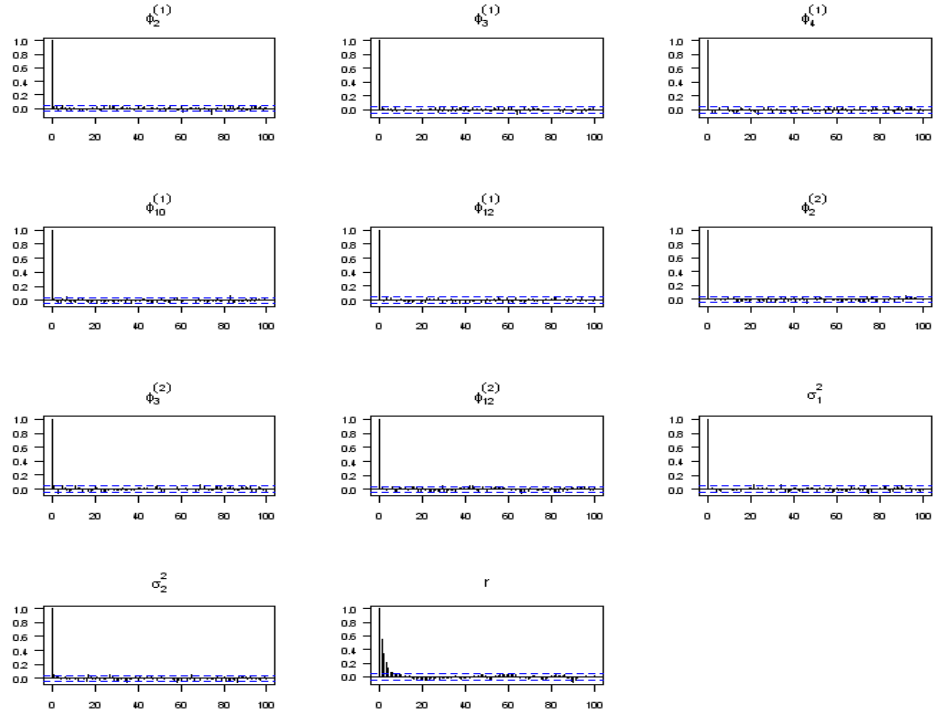

Figure 8: Autocorrelation plots of after burn-in MCMC iterations for all parameters.

C.W.S. Chen. A Bayesian analysis of generalized threshold autoregressive models. *Statistics and Probability Letters*, 40:15–22, 1998.

C.W.S. Chen and J.C. Lee. Bayesian inference of threshold autoregressive models. *J. Time Ser. Anal.*, 16:483–492, 1995.

A. Gelman, G.O. Roberts, and W.R. Gilks. Efficient Metropolis jumping rules. In: *Bayesian Statistics 5* (Edited by J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), 599–607, 1996. Oxford University Press, Oxford.

W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97-109, 1970.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth and A.H. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087-1091, 1953.

M.K.P. So and C.W.S. Chen. Subset threshold autoregression. *Journal of Forecasting*, 22, :49-66, 2003.

H. Tong. On a Threshold Model in Pattern Recognition and Signal Processing, ed. C. H. Chen, Sijhoff & Noordhoff: Amsterdam, 1978.

H. Tong. Threshold Models in Non-linear Time Series Analysis, Vol. 21 of Lecture Notes in Statistics (K. Krickegerg, ed.). Springer-Verlag, New York, 1983.

H. Tong and K.S. Lim. Threshold autoregression, limit cycles and cyclical data. (with discussion), *J. R. Stat. Soc. Ser. B*, 42:245–292, 1980.

R.S. Tsay. Testing and modeling threshold autoregressive process. *J. Amer. Statist. Assoc.*, 84:231–240, 1989.

R.S. Tsay. Testing and modeling multivariate threshold models. *J. Amer. Statist. Assoc.*, 93:1188–1202, 1998.

R.S. Tsay. Analysis of Financial Time Series, 2nd Edition, John Wiley & Sons, 2005.

Cathy W. S. Chen
Feng Chia University, Taiwan
chenws@fcu.edu.tw

Edward M. H. Lin, Feng Chi Liu
Feng Chia University, Taiwan

Richard Gerlach
University of Sydney, Australia
R.Gerlach@econ.usyd.edu.au

Statistical Modeling of Loss Distributions Using actuar

by Vincent Goulet and Mathieu Pigeon

Introduction

actuar (Dutang et al., 2008) is a package providing additional Actuarial Science functionality to R. Although various packages on CRAN provide functions useful to actuaries, **actuar** aims to serve as a central location for more specifically actuarial functions and data sets. The current feature set of the package can be split in four main categories: loss distributions modeling, risk theory (including ruin theory), simulation of compound hierarchical models and credibility theory.

This paper reviews the loss distributions modeling features of the package — those most likely to interest R News readers and to have links with other fields of statistical practice.

Actuaries need to model claim amounts distributions for ratemaking, loss reserving and other risk evaluation purposes. Typically, claim amounts data are nonnegative and skewed to the right, often heavily. The probability laws used in modeling must match these characteristics. Furthermore, depending on the line of business, data can be truncated from below, censored from above or both.

The main **actuar** features to aid in loss modeling are the following:

1. Introduction of 18 additional probability laws and functions to get raw moments, limited moments and the moment generating function.

2. Fairly extensive support of grouped data.

3. Calculation of the empirical raw and limited moments.

4. Minimum distance estimation using three different measures.

5. Treatment of coverage modifications (deductibles, limits, inflation, coinsurance).

Probability laws

R already includes functions to compute the probability density function (pdf), the cumulative distribution function (cdf) and the quantile function of a fair number of probability laws, as well as functions to generate variates from these laws. For some root `foo`, the functions are named `dfoo`, `pfoo`, `qfoo` and `rfoo`, respectively.

The **actuar** package provides `d`, `p`, `q` and `r` functions for all the probability laws useful for loss severity modeling found in Appendix A of Klugman et al. (2004) and not already present in base R, excluding the inverse Gaussian and log-t but including the loggamma distribution (Hogg and Klugman, 1984). We tried to make these functions as similar as possible to those in the **stats** package, with respect to the interface, the names of the arguments and the handling of limit cases.

Table 1 lists the supported distributions as named in Klugman et al. (2004) along with the root names of the R functions. The name or the parametrization of some distributions may differ in other fields; check with the `lossdist` package vignette for the pdf and cdf of each distribution.

In addition to the `d`, `p`, `q` and `r` functions, the package provides `m`, `lev` and `mgf` functions to compute, respectively, theoretical raw moments

$$m_k = \mathbb{E}[X^k], \tag{1}$$

theoretical limited moments

$$\mathbb{E}[(X \wedge x)^k] = \mathbb{E}[(\min X, x)^k] \tag{2}$$

and the moment generating function

$$M_X(t) = \mathbb{E}[e^{tX}], \tag{3}$$

when it exists. Every probability law of Table 1 is supported, plus the following ones: beta, exponential, chi-square, gamma, lognormal, normal (except `lev`), uniform and Weibull of base R, and the inverse Gaussian distribution of package **SuppDists** (Wheeler, 2006). The `m` and `lev` functions are especially useful with estimation methods based on the matching of raw or limited moments; see below for their empirical counterparts. The `mgf` functions are introduced in the package mostly for calculation of the adjustment coefficient in ruin theory; see the `"risk"` package vignette.

In addition to the 17 distributions of Table 1, the package provides support for phase-type distributions (Neuts, 1981). These are not so much included in the package for statistical inference, but rather for ruin probability calculations. A phase-type distribution is defined as the distribution of the time until absorption of a continuous time, finite state Markov process with m transient states and one absorbing state. Let

$$Q = \begin{bmatrix} \boldsymbol{T} & \boldsymbol{t} \\ \boldsymbol{0} & 0 \end{bmatrix} \tag{4}$$

be the transition rates matrix (or intensity matrix) of such a process and let $(\boldsymbol{\pi}, \pi_{m+1})$ be the initial probability vector. Here, $\boldsymbol{T}$ is an $m \times m$ non-singular matrix with $t_{ii} < 0$ for $i = 1, \ldots, m$ and $t_{ij} \geq 0$ for $i \neq j$,

Table 1: Probability laws supported by **actuar** classified by family and root names of the R functions.

Family	Distribution	Root (alias)
Transformed beta	Transformed beta	`trbeta` (`pearson6`)
	Burr	`burr`
	Loglogistic	`llogis`
	Paralogistic	`paralogis`
	Generalized Pareto	`genpareto`
	Pareto	`pareto` (`pareto2`)
	Inverse Burr	`invburr`
	Inverse Pareto	`invpareto`
	Inverse paralogistic	`invparalogis`
Transformed gamma	Transformed gamma	`trgamma`
	Inverse transformed gamma	`invtrgamma`
	Inverse gamma	`invgamma`
	Inverse Weibull	`invweibull` (`lgompertz`)
	Inverse exponential	`invexp`
Other	Loggamma	`lgamma`
	Single parameter Pareto	`pareto1`
	Generalized beta	`genbeta`

$\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e}$ and $\boldsymbol{e}$ is a column vector with all components equal to 1. Then the cdf of the time until absorption random variable with parameters $\boldsymbol{\pi}$ and $\boldsymbol{T}$ is

$$F(x) = \begin{cases} 1 - \boldsymbol{\pi} e^{\boldsymbol{T}x} \boldsymbol{e}, & x > 0 \\ \pi_{m+1}, & x = 0, \end{cases} \tag{5}$$

where

$$e^{\boldsymbol{M}} = \sum_{n=0}^{\infty} \frac{\boldsymbol{M}^n}{n!} \tag{6}$$

is the *matrix exponential* of matrix $\boldsymbol{M}$.

The exponential, the Erlang (gamma with integer shape parameter) and discrete mixtures thereof are common special cases of phase-type distributions. The package provides `d`, `p`, `r`, `m` and `mgf` functions for phase-type distributions. The root is `phtype` and parameters $\boldsymbol{\pi}$ and $\boldsymbol{T}$ are named `prob` and `rates`, respectively.

The core of all the functions presented in this section is written in C for speed. The matrix exponential C routine is based on `expm()` from the package **Matrix** (Bates and Maechler, 2007).

Grouped data

What is commonly referred to in Actuarial Science as grouped data is data represented in an interval-frequency manner. In insurance applications, a grouped data set will typically report that there were n_j claims in the interval $(c_{j-1}, c_j]$, $j = 1, \ldots, r$ (with the possibility that $c_r = \infty$). This representation is much more compact than an individual data set (where the value of each claim is known), but it also carries far less information. Now that storage space in computers has almost become a non-issue, grouped data has somewhat fallen out of fashion.

Still, grouped data remains useful in some fields of actuarial practice and for parameter estimation. For these reasons, **actuar** provides facilities to store, manipulate and summarize grouped data. A standard storage method is needed since there are many ways to represent grouped data in the computer: using a list or a matrix, aligning the n_js with the c_{j-1}s or with the c_js, omitting c_0 or not, etc. Moreover, with appropriate extraction, replacement and summary functions, manipulation of grouped data becomes similar to that of individual data.

First, function `grouped.data` creates a grouped data object similar to — and inheriting from — a data frame. The input of the function is a vector of group boundaries $c_0, c_1, \ldots, c_r$ and one or more vectors of group frequencies $n_1, \ldots, n_r$. Note that there should be one group boundary more than group frequencies. Furthermore, the function assumes that the intervals are contiguous. For example, the following data

Group	Frequency (Line 1)	Frequency (Line 2)
$(0, 25]$	30	26
$(25, 50]$	31	33
$(50, 100]$	57	31
$(100, 150]$	42	19
$(150, 250]$	65	16
$(250, 500]$	84	11

is entered and represented in R as

```
> x <- grouped.data(Group = c(0, 25,
+     50, 100, 150, 250, 500), Line.1 = c(30,
```

```
+     31, 57, 42, 65, 84), Line.2 = c(26,
+     33, 31, 19, 16, 11))
```

Object x is stored internally as a list with class

```
> class(x)

[1] "grouped.data" "data.frame"
```

With a suitable `print` method, these objects can be displayed in an unambiguous manner:

```
> x

       Group Line.1 Line.2
1   (0,  25]     30     26
2  (25,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11
```

Second, the package supports the most common extraction and replacement methods for `"grouped.data"` objects using the usual `[` and `[<-` operators; see `?Extract.grouped.data` for details.

The package defines methods of a few existing summary functions for grouped data objects. Computing the mean

$$\sum_{j=1}^{r} \left(\frac{c_{j-1} + c_j}{2} \right) n_j \Big/ \sum_{j=1}^{r} n_j \tag{7}$$

is made simple with a method for the `mean` function:

```
> mean(x)

Line.1 Line.2
 179.8   99.9
```

Higher empirical moments can be computed with `emm`; see below.

A method for function `hist` draws a histogram for already grouped data. Only the first frequencies column is considered (see Figure 1 for the resulting graph):

```
> hist(x[, -3])
```

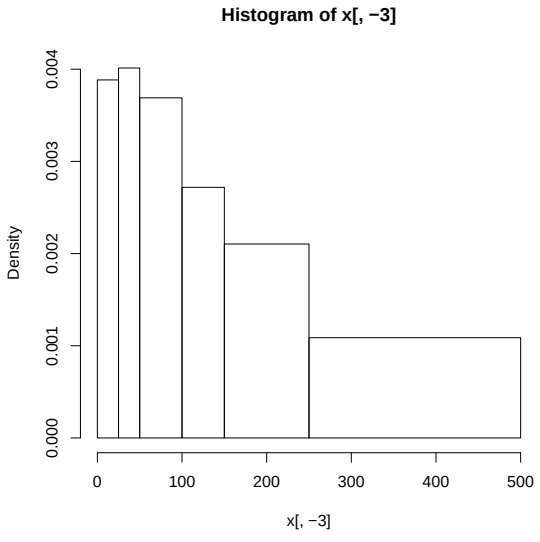

Figure 1: Histogram of a grouped data object

R has a function `ecdf` to compute the empirical cdf of an individual data set,

$$F_n(x) = \frac{1}{n} \sum_{j=1}^{n} I\{x_j \le x\},$$

where $I\{\mathcal{A}\} = 1$ if $\mathcal{A}$ is true and $I\{\mathcal{A}\} = 0$ otherwise. The function returns a `"function"` object to compute the value of $F_n(x)$ in any x. The approximation of the empirical cdf for grouped data is called an ogive (Klugman et al., 1998; Hogg and Klugman, 1984). It is obtained by joining the known values of $F_n(x)$ at group boundaries with straight line segments:

$$\tilde{F}_n(x) = \begin{cases} 0, & x \le c_0 \\ \dfrac{(c_j - x)F_n(c_{j-1}) + (x - c_{j-1})F_n(c_j)}{c_j - c_{j-1}}, & \\ \quad c_{j-1} < x \le c_j & \\ 1, & x > c_r. \end{cases} \tag{8}$$

The package includes a function `ogive` that otherwise behaves exactly like `ecdf`. In particular, methods for functions `knots` and `plot` allow, respectively, to obtain the knots $c_0, c_1, \ldots, c_r$ of the ogive and a graph.

Calculation of empirical moments

In the sequel, we frequently use two data sets provided by the package: the individual dental claims (`dental`) and grouped dental claims (`gdental`) of Klugman et al. (2004).

The package provides two functions useful for estimation based on moments. First, function `emm` computes the kth empirical moment of a sample, whether in individual or grouped data form:

```
> emm(dental, order = 1:3)

[1] 3.355e+02 2.931e+05 3.729e+08

> emm(gdental, order = 1:3)

[1] 3.533e+02 3.577e+05 6.586e+08
```

Second, in the same spirit as `ecdf` and `ogive`, function `elev` returns a function to compute the empirical limited expected value — or first limited moment — of a sample for any limit. Again, there are methods for individual and grouped data (see Figure 2 for the graphs):

```
> lev <- elev(dental)
> lev(knots(lev))

 [1]  16.0  37.6  42.4  85.1 105.5 164.5
 [7] 187.7 197.9 241.1 335.5

> plot(lev, type = "o", pch = 19)
> lev <- elev(gdental)
> lev(knots(lev))
```

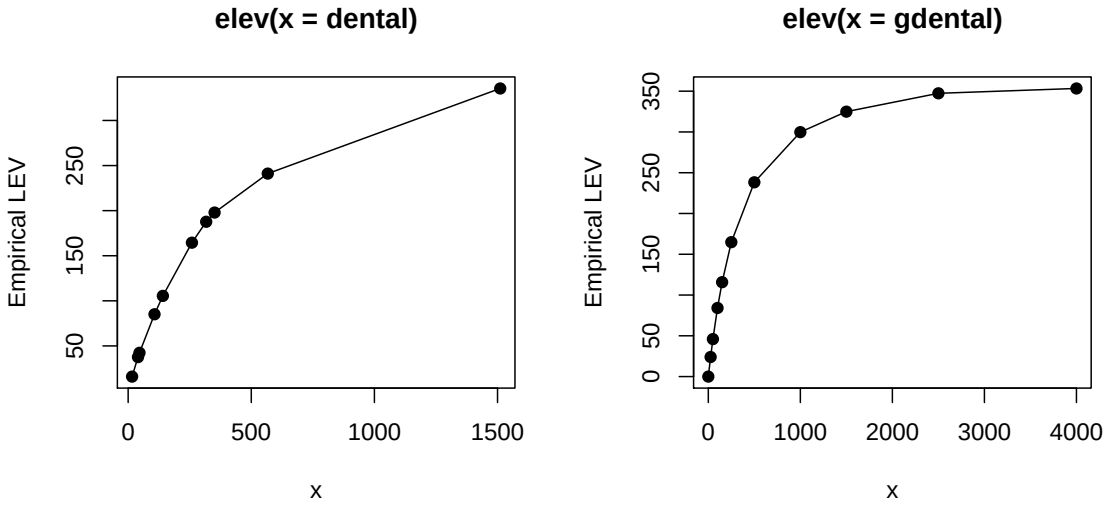

Figure 2: Empirical limited expected value function of an individual data object (left) and a grouped data object (right)

```
 [1]   0.00  24.01  46.00  84.16 115.77
 [6] 164.85 238.26 299.77 324.90 347.39
[11] 353.34

> plot(lev, type = "o", pch = 19)
```

Minimum distance estimation

Two methods are widely used by actuaries to fit models to data: maximum likelihood and minimum distance. The first technique applied to individual data is well covered by function `fitdistr` of the **MASS** package (Venables and Ripley, 2002).

The second technique minimizes a chosen distance function between theoretical and empirical distributions. The **actuar** package provides function `mde`, very similar in usage and inner working to `fitdistr`, to fit models according to any of the following three distance minimization methods.

1. The Cramér-von Mises method (`CvM`) minimizes the squared difference between the theoretical cdf and the empirical cdf or ogive at their knots:

$$d(\theta) = \sum_{j=1}^{n} w_j [F(x_j;\theta) - F_n(x_j;\theta)]^2 \qquad (9)$$

for individual data and

$$d(\theta) = \sum_{j=1}^{r} w_j [F(c_j;\theta) - \tilde{F}_n(c_j;\theta)]^2 \qquad (10)$$

for grouped data. Here, $F(x)$ is the theoretical cdf of a parametric family, $F_n(x)$ is the empirical cdf, $\tilde{F}_n(x)$ is the ogive and $w_1 \geq 0, w_2 \geq 0, \ldots$ are arbitrary weights (defaulting to 1).

2. The modified chi-square method (`chi-square`) applies to grouped data only and minimizes the squared difference between the expected and observed frequency within each group:

$$d(\theta) = \sum_{j=1}^{r} w_j [n(F(c_j;\theta) - F(c_{j-1};\theta)) - n_j]^2, \qquad (11)$$

where $n = \sum_{j=1}^{r} n_j$. By default, $w_j = n_j^{-1}$. The method is called "modified" because the default denominators are the observed rather than the expected frequencies.

3. The layer average severity method (`LAS`) applies to grouped data only and minimizes the squared difference between the theoretical and empirical limited expected value within each group:

$$d(\theta) = \sum_{j=1}^{r} w_j [\mathrm{LAS}(c_{j-1}, c_j;\theta) - \tilde{\mathrm{LAS}}_n(c_{j-1}, c_j;\theta)]^2, \qquad (12)$$

where $\mathrm{LAS}(x,y) = \mathbb{E}[X \wedge y] - \mathbb{E}[X \wedge x]$, $\tilde{\mathrm{LAS}}_n(x,y) = \tilde{\mathbb{E}}_n[X \wedge y] - \tilde{\mathbb{E}}_n[X \wedge x]$, and $\tilde{\mathbb{E}}_n[X \wedge x]$ is the empirical limited expected value for grouped data.

The arguments of `mde` are a data set, a function to compute $F(x)$ or $\mathbb{E}[X \wedge x]$, starting values for the optimization procedure and the name of the method to use. The empirical functions are computed with `ecdf`, `ogive` or `elev`.

The expressions below fit an exponential distribution to the grouped dental data set, as per Example 2.21 of Klugman et al. (1998):

```
> mde(gdental, pexp,
+     start = list(rate = 1/200),
+     measure = "CvM")

   rate
  0.003551

 distance
  0.002842

> mde(gdental, pexp,
+     start = list(rate = 1/200),
+     measure = "chi-square")

   rate
  0.00364

 distance
     13.54

> mde(gdental, levexp,
+     start = list(rate = 1/200),
+     measure = "LAS")

   rate
  0.002966

 distance
     694.5
```

It should be noted that optimization is not always that simple to achieve. For example, consider the problem of fitting a Pareto distribution to the same data set using the Cramér–von Mises method:

```
> mde(gdental, ppareto,
+     start = list(shape = 3,
+     scale = 600), measure = "CvM")

Error in mde(gdental, ppareto,
             start = list(shape = 3,
               scale = 600),
             measure = "CvM") :
  optimization failed
```

Working in the log of the parameters often solves the problem since the optimization routine can then flawlessly work with negative parameter values:

```
> f <- function(x, lshape, lscale) ppareto(x,
+     exp(lshape), exp(lscale))
> (p <- mde(gdental, f, list(lshape = log(3),
+     lscale = log(600)), measure = "CvM"))

 lshape   lscale
   1.581    7.128

  distance
  0.0007905
```

The actual estimators of the parameters are obtained with

```
> exp(p$estimate)

  lshape   lscale
   4.861 1246.485
```

This procedure may introduce additional bias in the estimators, though.

Coverage modifications

Let X be the random variable of the actual claim amount for an insurance policy, Y^L be the random variable of the amount paid per loss and Y^P be the random variable of the amount paid per payment. The terminology for the last two random variables refers to whether or not the insurer knows that a loss occurred. Now, the random variables X, Y^L and Y^P will differ if any of the following coverage modifications are present for the policy: an ordinary or a franchise deductible, a limit, coinsurance or inflation adjustment (see Klugman et al., 2004, Chapter 5 for precise definitions of these terms). Table 2 summarizes the definitions of Y^L and Y^P.

The effect of an ordinary deductible is known as truncation from below, and that of a policy limit as censoring from above. Censored data is very common in survival analysis; see the package **survival** (Lumley, 2008) for an extensive treatment in R. Yet, **actuar** provides a different approach.

Suppose one wants to use censored data $Y_1, \ldots, Y_n$ from the random variable Y to fit a model on the unobservable random variable X. This requires expressing the pdf or cdf of Y in terms of the pdf or cdf of X. Function `coverage` of **actuar** does just that: given a pdf or cdf and any combination of the coverage modifications mentioned above, `coverage` returns a function object to compute the pdf or cdf of the modified random variable. The function can then be used in modeling or plotting like any other `d`*`foo`* or `p`*`foo`* function.

For example, let Y represent the amount paid (per payment) by an insurer for a policy with an ordinary deductible d and a limit $u - d$ (or maximum covered loss of u). Then the definition of Y is

$$Y = \begin{cases} X - d, & d \leq X \leq u \\ u - d, & X \geq u \end{cases} \tag{13}$$

and its pdf is

$$f_Y(y) = \begin{cases} 0, & y = 0 \\ \dfrac{f_X(y+d)}{1 - F_X(d)}, & 0 < y < u - d \\ \dfrac{1 - F_X(u)}{1 - F_X(d)}, & y = u - d \\ 0, & y > u - d. \end{cases} \tag{14}$$

Assume X has a gamma distribution. Then an R function to compute the pdf (14) in any y for a deductible $d = 1$ and a limit $u = 10$ is obtained with `coverage` as follows:

```
> f <- coverage(pdf = dgamma, cdf = pgamma,
+     deductible = 1, limit = 10)
> f(0, shape = 5, rate = 1)
```

Table 2: Coverage modifications for per-loss variable (Y^L) and per-payment variable (Y^P) as defined in Klugman et al. (2004).

Coverage modification	Per-loss variable (Y^L)	Per-payment variable (Y^P)
Ordinary deductible (d)	$\begin{cases} 0, & X \leq d \\ X-d, & X > d \end{cases}$	$\begin{cases} X-d, & X > d \end{cases}$
Franchise deductible (d)	$\begin{cases} 0, & X \leq d \\ X, & X > d \end{cases}$	$\begin{cases} X, & X > d \end{cases}$
Limit (u)	$\begin{cases} X, & X \leq u \\ u, & X > u \end{cases}$	$\begin{cases} X, & X \leq u \\ u, & X > u \end{cases}$
Coinsurance (α)	αX	αX
Inflation (r)	$(1+r)X$	$(1+r)X$

```
[1] 0

> f(5, shape = 5, rate = 1)

[1] 0.1343

> f(9, shape = 5, rate = 1)

[1] 0.02936

> f(12, shape = 5, rate = 1)

[1] 0
```

The function `f` is built specifically for the coverage modifications submitted and contains as little useless code as possible.

Let object `y` contain a sample of claims amounts from policies with the above deductible and limit. Then one can fit a gamma distribution by maximum likelihood to the claim severity process as follows:

```
> library(MASS)
> fitdistr(y, f, start = list(shape = 2,
+     rate = 0.5))

   shape     rate
  4.1204   0.8230
 (0.7054) (0.1465)
```

The package vignette `"coverage"` contains more detailed pdf and cdf formulas under various combinations of coverage modifications.

Conclusion

This paper reviewed the main loss modeling features of **actuar**, namely many new probability laws; new utility functions to access the raw moments, the limited moments and the moment generating function of these laws and some of base R; functions to create and manipulate grouped data sets; functions to ease calculation of empirical moments, in particular for grouped data; a function to fit models by distance minimization; a function to help work with censored data or data subject to coinsurance or inflation adjustments.

We hope some of the tools presented here may be of interest outside the field they were developed for, perhaps provided some adjustments in terminology and nomenclature.

Finally, we note that the **distr*XXX*** family of packages (Ruckdeschel et al., 2006) provides a general, object oriented approach to some of the features of **actuar**, most notably the calculation of moments for many distributions (although not necessarily those presented here), minimum distance estimation and censoring.

Acknowledgments

This research benefited from financial support from the Natural Sciences and Engineering Research Council of Canada and from the *Chaire d'actuariat* (Actuarial Science Chair) of Université Laval. We also thank one anonymous referee and the R News editors for corrections and improvements to the paper.

Bibliography

D. Bates and M. Maechler. **Matrix***: A matrix package for R*, 2007. R package version 0.999375-3.

C. Dutang, V. Goulet, and M. Pigeon. **actuar**: An R package for actuarial science. *Journal of Statistical Software*, 25(7), 2008. URL `http://www.actuar-project.org`.

R. V. Hogg and S. A. Klugman. *Loss Distributions*. Wiley, New York, 1984. ISBN 0-4718792-9-0.

S. A. Klugman, H. H. Panjer, and G. Willmot. *Loss Models: From Data to Decisions*. Wiley, New York, 1998. ISBN 0-4712388-4-8.

S. A. Klugman, H. H. Panjer, and G. Willmot. *Loss Models: From Data to Decisions*. Wiley, New York, 2 edition, 2004. ISBN 0-4712157-7-5.

T. Lumley. **survival***: Survival analysis, including penalised likelihood*, 2008. R package version 2.34. S original by Terry Therneau.

M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Dover Publications, 1981. ISBN 978-0-4866834-2-3.

P. Ruckdeschel, M. Kohl, T. Stabla, and F. Camphausen. S4 classes for distributions. *R News*, 6 (2):2–6, May 2006. URL `http://distr.r-forge.r-project.org`.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 4 edition, 2002. ISBN 0-3879545-7-0.

B. Wheeler. **SuppDists***: Supplementary distributions*, 2006. URL `http://www.bobwheeler.com/stat`. R package version 1.1-0.

Vincent Goulet and Mathieu Pigeon
Université Laval, Canada
`vincent.goulet@act.ulaval.ca`

Programmers' Niche: Multivariate polynomials in R

The multipol package

by Robin K. S. Hankin

Abstract

In this short article I introduce the **multipol** package, which provides some functionality for handling multivariate polynomials; the package is discussed here from a programming perspective. An example from the field of enumerative combinatorics is presented.

Univariate polynomials

A *polynomial* is an algebraic expression of the form $\sum_{i=0}^{n} a_i x^i$ where the a_i are real or complex numbers and n (the *degree* of the polynomial) is a nonnegative integer. A polynomial may be viewed in three distinct ways:

- Polynomials are interesting and instructive examples of entire functions: they map $\mathbb{C}$ (the complex numbers) to $\mathbb{C}$.
- Polynomials are a map from the positive integers to $\mathbb{C}$: this is $f(n) = a_n$ and one demands that $\exists n_0$ with $n \geqslant n_0 \longrightarrow f(n) = 0$. Relaxation of the final clause results in a *generating function* which is useful in combinatorics.
- Polynomials with complex coefficients form an algebraic object known as a *ring*: polynomial multiplication is associative and distributive with respect to addition; $(ab)c = a(bc)$ and $a(b+c) = ab + ac$.

A *multivariate polynomial* is a generalization of a polynomial to expressions of the form $\sum a_{i_1 i_2 \ldots i_d} \prod_{j=1}^{d} x_j^{i_j}$. The three characterizations of polynomials above generalize to the multivariate case, but note that the algebraic structure is more general.

In the context of R programming, the first two points typically dominate. Viewing a polynomial as a function is certainly second nature to the current readership and unlikely to yield new insight. But generating functions are also interesting and useful applications of polynomials (Wilf, 1994) which may be less familiar and here I discuss an example from the discipline of integer partitions (Andrews, 1998).

A *partition of an integer* n is a non-increasing sequence of positive integers $p_1, p_2, \ldots, p_r$ such that $n = \sum_{i=1}^{r} p_i$ (Hankin, 2006b). How many distinct partitions does n have?

The answer is the coefficient of x^n in

$$\prod_{i=1}^{n} \frac{1}{1-x^i}$$

(observe that we may truncate the Taylor expansion of $1/(1-x^j)$ to terms not exceeding x^n; thus the problem *is* within the domain of polynomials as infinite sequences of coefficients are not required). Here, as in many applications of generating functions, one uses the mechanism of polynomial multiplication as a bookkeeping device to keep track of the possibilities. The R idiom used in the **polynom** package is a spectacularly efficient method for doing so.

Multivariate polynomials generalize the concept of generating function, but in this case the functions are from n-tuples of nonnegative integers to $\mathbb{C}$. An example is given in the appendix below.

The polynom package

The **polynom** package (Venables et al., 2007) is a consistent and convenient suite of software for manipulating polynomials. This package was originally written in 1993 and is used by Venables and Ripley (2001) as an example of S3 classes.

The following R code shows the **polynom** package in use; the examples are then generalized to the multivariate case using the **multipol** package.

```
> require(polynom)
> (p <- polynomial(c(1, 0, 0, 3, 4)))

1 + 3*x^3 + 4*x^4

> str(p)

Class 'polynomial'  num [1:5] 1 0 0 3 4
```

See how a polynomial is represented as a vector of coefficients with `p[i]` holding the coefficient of x^{i-1}; note the off-by-one issue. Observe the natural print method which suppresses the zero entries—but the internal representation requires all coefficients so a length 5 vector is needed to store the object.

Polynomials may be multiplied and added:

```
> p + polynomial(1:2)

2 + 2*x + 3*x^3 + 4*x^4

> p * p

1 + 6*x^3 + 8*x^4 + 9*x^6 + 24*x^7 + 16*x^8
```

 ISSN 1609-3631

Note the overloading of '+' and '*': polynomial addition and multiplication are executed using the natural syntax on the command line. Observe that the addition is not entirely straightforward: the shorter polynomial must be padded with zeros.

A polynomial may be viewed either as an object, or a function. Coercing a polynomial to a function is straightforward:

```
> f1 <- as.function(p)
> f1(pi)

[1] 483.6552

> f1(matrix(1:6, 2, 3))

     [,1] [,2] [,3]
[1,]    8  406 2876
[2,]   89 1217 5833
```

Note the effortless and transparent vectorization of `f1()`.

Multivariate polynomials

There exist several methods by which polynomials may be generalized to multipols. To this author, the most natural is to consider an array of coefficients; the dimensionality of the array corresponds to the arity of the multipol. However, other methods suggest themselves and a brief discussion is given at the end.

Much of the univariate polynomial functionality presented above is directly applicable to multivariate polynomials.

```
> require(multipol)
> (a <- as.multipol(matrix(1:10, nrow = 2)))

    y^0 y^1 y^2 y^3 y^4
x^0   1   3   5   7   9
x^1   2   4   6   8  10
```

See how a multipol is actually an array, with one extent per variable present, in this case 2, although the package is capable of manipulating polynomials of arbitrary arity.

Multipol addition is a slight generalization of the univariate case:

```
> b <- as.multipol(matrix(1:10, ncol = 2))
> a + b

    y^0 y^1 y^2 y^3 y^4
x^0   2   9   5   7   9
x^1   4  11   6   8  10
x^2   3   8   0   0   0
x^3   4   9   0   0   0
x^4   5  10   0   0   0
```

In the multivariate case, the zero padding must be done in each array extent; the natural command-line syntax is achieved by defining an appropriate `Ops.multipol()` function to overload the arithmetic operators.

Multivariate polynomial multiplication

The heart of the package is multipol multiplication:

```
> a * b

    y^0 y^1 y^2 y^3 y^4 y^5
x^0   1   9  23  37  51  54
x^1   4  29  61  93 125 123
x^2   7  39  79 119 159 142
x^3  10  49  97 145 193 161
x^4  13  59 115 171 227 180
x^5  10  40  70 100 130 100
```

Multivariate polynomial multiplication is considerably more involved than in the univariate case. Consider the coefficient of x^2y^2 in the product. This is

$$\begin{aligned}
&C_a\left(x^2y^2\right)C_b\left(1\right) + C_a\left(xy^2\right)C_b\left(x\right) + C_a\left(y^2\right)C_b\left(x^2\right)\\
&+ C_a\left(x^2y\right)C_b\left(y\right) + C_a\left(xy\right)C_b\left(xy\right) + C_a\left(y\right)C_b\left(x^2y\right)\\
&+ C_a\left(x^2\right)C_b\left(y^2\right) + C_a\left(x\right)C_b\left(xy^2\right) + C_a\left(1\right)C_b\left(x^2y^2\right)\\
=\;& 0\cdot 1 + 6\cdot 2 + 5\cdot 3\\
&+ 0\cdot 6 + 4\cdot 7 + 3\cdot 8\\
&+ 0\cdot 0 + 2\cdot 0 + 1\cdot 0\\
=\;& 79,
\end{aligned}$$

where "$C_a\left(x^my^n\right)$" means the coefficient of x^my^n in polynomial `a`. It should be clear that large multipols involve more terms and a typical example is given later in the paper.

Multivariate polynomial multiplication in multipol

The appropriate R idiom is to follow the above prose description in a vectorized manner; the following extract from `mprod()` is very slightly edited in the interests of clarity.

First we define a matrix, `index`, whose rows are the array indices of the product:

```
outDims <- dim(a)+dim(b)-1
```

Here `outDims` *is the dimensions of the product. Note again the off-by-one issue: the package uses array indices internally, while the user consistently indexes by variable power.*

```
index <- expand.grid(lapply(outDims,seq_len))
```

Each row of matrix `index` *is thus an array index for the product.*

The next step is to define a convenience function `f()`*, whose argument* `u` *is a row of* `index`*, that returns the entry in the multipol product:*

```
f <- function(u){
  jja <-
  expand.grid(lapply(u,function(i)0:(i-1)))
  jjb <- -sweep(jja, 2, u)-1
```

So `jja` *is the (power) index of* `a`*, and the rows of* `jjb` *added to those of* `jja` *give* `u`*, which is the power index of the returned array. Now not all rows of* `jja` *and* `jjb` *correspond to extant elements of* `a` *and* `b` *respectively; so define a Boolean variable* `wanted` *that selects just the appropriate rows:*

```
wanted <-
apply(jja,1,function(x)all(x < dim(a))) &
apply(jjb,1,function(x)all(x < dim(b))) &
apply(jjb,1,function(x)all(x >= 0))
```

Thus element `n` *of* `wanted` *is* `TRUE` *only if the* `n`*th row of both* `jja` *and* `jjb` *correspond to a legal element of* `a` *and* `b` *respectively. Now perform the addition by summing the products of the legal elements:*

```
sum(a[1+jja[wanted,]] * b[1+jjb[wanted,]])
}
```

Thus function `f()` *returns the coefficient, which is the sum of products of pairs of legal elements of* `a` *and* `b`*. Again observe the off-by-one issue.*

Now `apply()` *function* `f()` *to the rows of* `index` *and reshape:*

```
out <- apply(index,1,f)
dim(out) <- outDims
```

Thus array `out` *contains the multivariate polynomial product of* `a` *and* `b`*.*

The preceding code shows how multivariate polynomials may be multiplied. The implementation makes no assumptions about the entries of `a` or `b` and the coefficients of the product are summed over all possibilities; opportunities to streamline the procedure are discussed below.

Multipols as functions

Polynomials are implicitly functions of one variable; multivariate polynomials are functions too, but of more than one argument. Coercion of a multipol to a function is straightforward:

```
> f2 <- as.function(a * b)

> f2(c(x = 1, y = 0+3i))

[1] 67725+167400i
```

It is worth noting the seamless integration between **polynom** and **multipol** in this regard: `f1(a)` is a multipol [recall that `f1()` is a function coerced from a univariate polynomial].

Multipol extraction and replacement

One often needs to extract or replace parts of a multipol. The package includes extraction and replacement methods but, partly because of the off-by-one issue, these are not straightforward.

Consider the case where one has a multipol and wishes to extract the terms of order zero ane one:

```
> a[0:1, 0:1]

     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Note how the off-by-one issue is handled: `a[i,j]` is the coefficient of $x^i y^j$ (here the constant and first-order terms); the code is due to Rougier (2007). Replacement is slightly different:

```
> a[0, 0] <- -99
> a

    y^0 y^1 y^2 y^3 y^4
x^0 -99   3   5   7   9
x^1   2   4   6   8  10
```

Observe how replacement operators—unlike extraction operators—return a multipol; this allows expeditious modification of multivariate polynomials. The reason that the extraction operator returns an array rather than a multipol is that the extracted object often does not have unambiguous interpretation as a multipol (consider `a[-1,-1]`, for example). It seems to this author that the loss of elegance arising from the asymmetry between extraction and replacement is amply offset by the impossibility of an extracted object's representation as a multipol being undesired—unless the user explicitly coerces.

The elephant in the room

Representing a multivariate polynomial by an array is a natural and efficient method, but suffers some disadvantages.

Consider Euler's four-square identity

$$\begin{aligned}\left(a_1^2+a_2^2+a_3^2+a_4^2\right)\cdot\left(b_1^2+b_2^2+b_3^2+b_4^2\right) = \\ (a_1b_1-a_2b_2-a_3b_3-a_4b_4)^2+ \\ (a_1b_2+a_2b_1+a_3b_4-a_4b_3)^2+ \\ (a_1b_3-a_2b_4+a_3b_1+a_4b_2)^2+ \\ (a_1b_4+a_2b_3-a_3b_2+a_4b_1)^2\end{aligned}$$

which was discussed in 1749 in a letter from Euler to Goldbach. The identity is important in number

[1] Or indeed more elegantly by observing that both sides of the identity express the absolute value of the product of two quaternions: $|a|^2\,|b|^2 = |ab|^2$. With the **onion** package (Hankin, 2006a), one would define `f <- function(a,b)Norm(a)*Norm(b) - Norm(a*b)` and observe (for example) that `f(rquat(rand="norm"),rquat(rand="norm"))` is zero to machine precision.

theory, and may be proved straightforwardly by direct expansion[1]. It may by verified to machine precision using the **multipol** package; the left hand side is given by:

```
> options("showchars" = TRUE)
> lhs <- polyprod(ones(4,2),ones(4,2))

[1] "1*x1^2*x5^2 + 1*x2^2*x5^2 + ...
```

(the right hand side's idiom is more involved), but this relatively trivial expansion requires about 20 minutes on my 1.5 GHz G4; the product comprises $3^8 = 6561$ elements, of which only 16 are nonzero. Note the `options()` statement controlling the format of the output which causes the result to be printed in a more appropriate form. Clearly the **multipol** package as currently implemented is inefficient for multivariate problems of this nature in which the arrays possess few nonzero elements.

A challenge

The inefficiency discussed above is ultimately due to the storage and manipulation of many zero coefficients that may be omitted from a calculation. Multivariate polynomials for which this is an issue appear to be common: the package includes many functions—such as `uni()`, `single()`, and `lone()`—that define useful multipols in which the number of nonzero elements is very small.

In this section, I discuss some ideas for implementations in which zero operations are implicitly excluded. These ideas are presented in the spirit of a request for comments: although they seem to this author to be reasonable methodologies, readers are invited to discuss the ideas presented here and indeed to suggest alternative strategies.

The canonical solution would be to employ some form of sparse array class, along the lines of Mathematica's `SparseArray`. Unfortunately, no such functionality exists as of 2008, but C++ includes a "map" class (Stroustrup, 1997) that would be ideally suited to this application.

There are other paradigms that may be worth exploring. It is possible to consider a multivariate polynomial of arity d (call this an object of class P^d) as being a univariate polynomial whose coefficients are of class P^{d-1}—class P^0 would be a real or complex number—but such recursive class definitions appear not to be possible with the current implementation of S3 or S4 (Venables, 2008). Recent experimental work by West (2008) exhibits a proof-of-concept in C++ which might form the back end of an R implementation. Euler's identity appears to be a particularly favourable example and is proved essentially instantaneously (the proof is a rigorous theoretical result, not just a numerical verification, as the system uses exact integer arithmetic).

Conclusions

This article introduces the **multipol** package that provides functionality for manipulating multivariate polynomials. The **multipol** package builds on and generalizes the **polynom** package of Venables et al., which is restricted to the case of univariate polynomials. The generalization is not straightforward and presents a number of programming issues that were discussed.

One overriding issue is that of performance: many multivariate polynomials of interest are "sparse" in the sense that they have many zero entries that unnecessarily consume storage and processing resources.

Several possible solutions are suggested, in the form of a request for comments. The canonical method appears to be some form of sparse array, for which the "map" class of the C++ language is ideally suited. Implementation of such functionality in R might well find application in fields other than multivariate polynomials.

Appendix: an example

This appendix presents a brief technical example of multivariate polynomials in use in the field of enumerative combinatorics (Good, 1976). Suppose one wishes to determine how many contingency tables, with non-negative integer entries, have specified row and column marginal totals. The appropriate generating function is

$$\prod_{1\leqslant i\leqslant nr}\prod_{1\leqslant j\leqslant nc}\frac{1}{1-x_i y_j}$$

where the table has nr rows and nc columns (the number of contingency tables is given by the coefficient of $x_1^{s_1}x_2^{s_2}\cdots x_r^{s_r}\cdot y_1^{t_1}y_2^{t_2}\cdots y_t^{t_c}$ where the s_i and t_i are the row- and column- sums respectively). The R idiom for the generating function `gf` in the case of $nr = nc = n = 3$ is:

```
 n <- 3
jj <- as.matrix(expand.grid(1:n,n+(1:n)))
 f <- function(i) ooom(n,lone(2*n,jj[i,]),m=n)
 u <- c(sapply(1:(n*n),f,simplify=FALSE))
gf <- do.call("mprod", c(u,maxorder=n))
```

[here function `ooom()` is "one-over-one-minus"; and `mprod()` is the function name for multipol product]. In this case, it is clear that sparse array functionality would not result in better performance, many elements of the generating function `gf` are nonzero. Observe that the maximum of `gf`, 55, is consistent with Sloane (2008).

Acknowledgements

I would like to acknowledge the many stimulating comments made by the R-help list. In particular,

the insightful comments from Bill Venables and Kurt Hornik were extremely helpful.

Bibliography

G. E. Andrews. *The Theory of Partitions*. Cambridge University Press, 1998.

L. Euler. Lettre CXXV. Communication to Goldbach; Berlin, 12 April, 1749.

I. J. Good. On the application of symmetric Dirichlet distributions and their mixtures to contingency tables. *The Annals of Statistics*, 4(6):1159–1189, 1976.

R. K. S. Hankin. Normed division algebras with R: Introducing the **onion** package. *R News*, 6(2): 49–52, May 2006a. URL `http://CRAN.R-project.org/doc/Rnews/`.

R. K. S. Hankin. Additive integer partitions in R. *Journal of Statistical Software, Code Snippets*, 16(1), May 2006b.

J. Rougier. **Oarray***: Arrays with arbitrary offsets*, 2007. R package version 1.4-2.

N. J. A. Sloane. The on-line encyclopedia of integer sequences. Published electronically at `http://www.research.att.com/~njas/sequences/A110058`, 2008.

B. Stroustrup. *The C++ Programming Language*. Addison Wesley, third edition, 1997.

W. N. Venables, K. Hornik, and M. Maechler. **polynom***: A collection of functions to implement a class for univariate polynomial manipulations*, 2007. URL `http://CRAN.R-project.org/`. R package version 1.3-2. S original by Bill Venables, packages for R by Kurt Hornik and Martin Maechler.

W. N. Venables. Personal Communication, 2008.

W. N. Venables and B. D. Ripley. *S Programming*. Springer, 2001.

L. J. West. An experimental C++ implementation of a recursively defined polynomial class. Personal communication, 2008.

H. S. Wilf. *generatingfunctionology*. Academic Press, 1994.

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

R Help Desk

How Can I Avoid This Loop or Make It Faster?

by Uwe Ligges and John Fox

Introduction

There are some circumstances in which optimized and efficient code is desirable: in functions that are frequently used, in functions that are made available to the public (e.g., in a package), in simulations taking a considerable amount of time, etc. There are other circumstances in which code should *not* be optimized with respect to speed — if the performance is already satisfactory. For example, in order to save a few seconds or minutes of CPU time, you do not want to spend a few hours of programming, and you do not want to break code or introduce bugs applying optimization patches to properly working code.

A principal rule is: Do not optimize unless you really need optimized code! Some more thoughts about this rule are given by Hyde (2006), for example. A second rule in R is: When you write a function from scratch, do it the vectorized way initially. If you do, then most of the time there will be no need to optimize later on.

If you really need to optimize, measure the speed of your code rather than guessing it. How to *profile* R code in order to detect the bottlenecks is described in Venables (2001), R Development Core Team (2008a), and the help page `?Rprof`. The CRAN packages **proftools** (Tierney, 2007) and **profr** (Wickham, 2008) provide sets of more extensive profiling tools.

The convenient function `system.time()` (used later in this article) simply measures the time of the command given as its argument.[1] The returned value consists of user time (CPU time R needs for calculations), system time (time the system is using for processing requests, e.g., for handling files), total time (how long it really took to process the command) and — depending on the operating system in use — two further elements.

Readability and clarity of the code is another topic in the area of optimized code that has to be considered, because readable code is more maintainable, and users (as well as the author) can easily see what is going on in a particular piece of code.

In the next section, we focus on vectorization to optimize code both for speed and for readability. We describe the use of the family of `*apply` functions, which enable us to write condensed but clear code. Some of those functions can even make the code perform faster. How to avoid mistakes when writing loops and how to measure the speed of code is described in a subsequent section.

Vectorization!

R is an interpreted language, i.e., code is parsed and evaluated at runtime. Therefore there is a speed issue which can be addressed by writing vectorized code (which is executed vector-wise) rather than using loops, if the problem can be vectorized. Loops are not necessarily bad, however, if they are used in the right way — and if some basic rules are heeded: see the section below on loops.

Many vector-wise operations are obvious. Nobody would want to replace the common component-wise operators for vectors or matrices (such as `+`, `-`, `*`, ...), matrix multiplication (`%*%`), and extremely handy vectorized functions such as `crossprod()` and `outer()` by loops. Note that there are also very efficient functions available for calculating sums and means for certain dimensions in arrays or matrices: `rowSums()`, `colSums()`, `rowMeans()`, and `colMeans()`.

If vectorization is not as obvious as in the cases mentioned above, the functions in the 'apply' family, named `[s,l,m,t]apply`, are provided to apply another function to the elements/dimensions of objects. These 'apply' functions provide a compact syntax for sometimes rather complex tasks that is more readable and faster than poorly written loops.

Matrices and arrays: `apply()`

The function `apply()` is used to work vector-wise on matrices or arrays. Appropriate functions can be applied to the columns or rows of a matrix or array without explicitly writing code for a loop. Before reading further in this article, type `?apply` and read the whole help page, particularly the sections 'Usage', 'Arguments', and 'Examples'.

As an example, let us construct a 5×4 matrix `X` from some random numbers (following a normal distribution with $\mu = 0, \sigma = 1$) and apply the function `max()` column-wise to `X`. The result will be a vector of the maxima of the columns:

```
R> (X <- matrix(rnorm(20), nrow = 5, ncol = 4))
R> apply(X, 2, max)
```

Dataframes, lists and vectors: `lapply()` and `sapply()`

Using the function `lapply()` (*l* because the value returned is a *l*ist), another appropriate function can be quickly applied element-wise to other objects, for example, dataframes, lists, or simply vectors. The resulting list has as many elements as the original object to which the function is applied.

[1] Timings in this article have been measured on the following platform: AMD Athlon 64 X2 Dual Core 3800+ (2 GHz), 2 Gb RAM, Windows XP Professional SP2 (32-bit), using an optimized 'Rblas.dll' linked against ATLAS as available from CRAN.

Analogously, the function `sapply()` (*s* for *s*implify) works like `lapply()` with the exception that it tries to simplify the value it returns. This means, for example, that if the resulting object is a list containing just vectors of length one, the result simplifies to a vector (or a matrix, if the list contains vectors of equal lengths). If `sapply()` cannot simplify the result, it returns the same list as `lapply()`.

A frequently used R idiom: Suppose that you want to extract the *i*-th columns of several matrices that are contained in a list `L`. To set up an example, we construct a list `L` containing two matrices `A` and `B`:

```
R> A <- matrix(1:4, 2, 2)
R> B <- matrix(5:10, 2, 3)
R> (L <- list(A, B))
    [[1]]
        [,1] [,2]
    [1,]    1    3
    [2,]    2    4

    [[2]]
        [,1] [,2] [,3]
    [1,]    5    7    9
    [2,]    6    8   10
```

The next call can be read as follows: 'Apply the function `[()` to all elements of `L` as the first argument, omit the second argument, and specify 2 as the third argument. Finally return the result in the form of a list.' The command returns the second columns of both matrices in the form of a list:

```
R> lapply(L, "[", , 2)
    [[1]]
    [1] 3 4

    [[2]]
    [1] 7 8
```

The same result can be achieved by specifying an anonymous function, as in:

```
R> sapply(L, function(x) x[ , 2])
        [,1] [,2]
    [1,]    3    7
    [2,]    4    8
```

where the elements of `L` are passed separately as `x` in the argument of the anonymous function given as the second argument in the `lapply()` call. Because all matrices in `L` contain equal numbers of rows, the call returns a matrix consisting of the second columns of all the matrices in `L`.

Vectorization via `mapply()` and `Vectorize()`

The `mapply()` function (*m* for *m*ultivariate) can simultaneously vectorize several arguments to a function that does not normally take vector arguments. Consider the `integrate()` function, which approximates definite integrals by adaptive quadrature, and which is designed to compute a single integral. The following command, for example, integrates the standard-normal density function from -1.96 to 1.96:

```
R> integrate(dnorm, lower=-1.96, upper=1.96)
   0.9500042 with absolute error < 1.0e-11
```

`integrate()` returns an object, the first element of which, named `"value"`, contains the value of the integral. This is an artificial example because normal integrals can be calculated more directly with the vectorized `pnorm()` function:

```
> pnorm(1.96) - pnorm(-1.96)
[1] 0.9500042
```

`mapply()` permits us to compute several normal integrals simultaneously:

```
R> (lo <- c(-Inf, -3:3))
   [1] -Inf   -3   -2   -1    0    1    2    3
R> (hi <- c(-3:3, Inf))
   [1]  -3  -2  -1   0   1   2   3 Inf

R> (P <- mapply(function(lo, hi)
 +   integrate(dnorm, lo, hi)$value, lo, hi))
   [1] 0.001349899 0.021400234 0.135905122
   [4] 0.341344746 0.341344746 0.135905122
   [7] 0.021400234 0.001349899

R> sum(P)
   [1] 1
```

`vectorize()` takes a function as its initial argument and returns a vectorized version of the function. For example, to vectorize `integrate()`:

```
R> Integrate <- Vectorize(
 +   function(fn, lower, upper)
 +     integrate(fn, lower, upper)$value,
 +   vectorize.args=c("lower", "upper")
 + )
```

Then

```
R> Integrate(dnorm, lower=lo, upper=hi)
```

produces the same result as the call to `mapply()` above.

Optimized BLAS for vectorized code

If vector and matrix operations (such as multiplication, inversion, decomposition) are applied to very large matrices and vectors, optimized BLAS (Basic Linear Algebra Subprograms) libraries can be used in order to increase the speed of execution dramatically, because such libraries make use of the specific architecture of the CPU (optimally using caches, pipelines, internal commands and units of a CPU). A well known optimized BLAS is ATLAS (Automatically Tuned Linear Algebra Software, `http://math-atlas.sourceforge.net/`, Whaley and Petitet, 2005). How to link R against ATLAS, for example, is discussed in R Development Core Team (2008b).

Windows users can simply obtain precompiled binary versions of the file 'Rblas.dll', linked against ATLAS for various CPUs, from the directory

'/bin/windows/contrib/ATLAS/' on their favourite CRAN mirror. All that is necessary is to replace the standard file 'Rblas.dll' in the 'bin' folder of the R installation with the file downloaded from CRAN. In particular, it is not necessary to recompile R to use the optimized 'Rblas.dll'.

Loops!

Many comments about R state that using loops is a particularly bad idea. This is not necessarily true. In certain cases, it is difficult to write vectorized code, or vectorized code may consume a huge amount of memory. Also note that it is in many instances much better to solve a problem with a loop than to use recursive function calls.

Some rules for writing loops should be heeded, however:

Initialize new objects to full length before the loop, rather than increasing their size within the loop.

If an element is to be assigned into an object in each iteration of a loop, and if the final length of that object is known before the loop starts, then the object should be initialized to full length prior to the loop. Otherwise, memory has to be allocated and data has to be copied in each iteration of the loop, which can take a considerable amount of time.

To initialize objects we can use functions such as

- `logical()`, `integer()`, `numeric()`, `complex()`, and `character()` for vectors of different modes, as well as the more general function `vector()`;
- `matrix()` and `array()`.

Consider the following example. We write three functions, `time1()`, `time2()`, and `time3()`, each assigning values element-wise into an object: For $i = 1, \ldots, n$, the value i^2 will be written into the i-th element of vector `a`. In function `time1()`, `a` will not be initialized to full length (very bad practice, but we see it repeatedly: `a <- NULL`):

```
R> time1 <- function(n){
 +     a <- NULL
 +     for(i in 1:n) a <- c(a, i^2)
 +     a
 + }
R> system.time(time1(30000))
   user  system elapsed
   5.11    0.01    5.13
```

In function `time2()`, `a` will be initialized to full length [`a <- numeric(n)`]:

```
R> time2 <- function(n){
 +     a <- numeric(n)
 +     for(i in 1:n) a[i] <- i^2
 +     a
 + }
R> system.time(time2(30000))
   user  system elapsed
   0.22    0.00    0.22
```

In function `time3()`, `a` will be created by a vectorwise operation without a loop.

```
R> time3 <- function(n){
 +     a <- (1:n)^2
 +     a
 + }
R> system.time(time3(30000))
   user  system elapsed
      0       0       0
```

What we see is that

- it makes sense to measure and to think about speed;
- functions of similar length of code and with the same results can vary in speed — drastically;
- the fastest way is to use a vectorized approach [as in `time3()`]; and
- if a vectorized approach does not work, remember to initialize objects to full length as in `time2()`, which was in our example more than 20 times faster than the approach in `time1()`.

A subtle point is that the advice to initialize objects applies to "atomic" objects, not to lists, where initialization is not advantageous. We invite readers to try the following code (which pertains to an example that we develop below):

```
R> system.time({
 +   matrices <- vector(mode="list", length=1000)
 +   for (i in seq_along(matrices))
 +     matrices[[i]] <-
 +       matrix(rnorm(10000), 100, 100)
 + })

R> system.time({
 +   matrices <- list()
 +   for (i in 1:1000)
 +     matrices[[i]] <-
 +       matrix(rnorm(10000), 100, 100)
 + })
```

Notice, however, that if you deliberately build up the object as you go along, it will slow things down a great deal, as the entire object will be copied at every step. Compare both of the above with the following:

```
R> system.time({
 +   matrices <- list()
 +   for (i in 1:1000)
 +     matrices <- c(matrices,
 +      list(matrix(rnorm(10000), 100, 100)))
 + })
```

Do not do things in a loop that can be done outside the loop.

It does not make sense, for example, to check for the validity of objects within a loop if checking can be applied outside, perhaps even vectorized.

It also does not make sense to apply the same calculations several times, particularly not n times

within a loop, if they just have to be performed one time.

Consider the following example where we want to apply a function [here `sin()`] to $i = 1, \ldots, n$ and multiply the results by 2π. Let us imagine that this function cannot work on vectors [although `sin()` does work on vectors, of course!], so that we need to use a loop:

```
R> time4 <- function(n){
 +     a <- numeric(n)
 +     for(i in 1:n)
 +         a[i] <- 2 * pi * sin(i)
 +     a
 + }
R> system.time(time4(100000))
   user  system elapsed
   0.75    0.00    0.75

R> time5 <- function(n){
 +     a <- numeric(n)
 +     for(i in 1:n)
 +         a[i] <- sin(i)
 +     2 * pi * a
 + }
R> system.time(time5(100000))
   user  system elapsed
   0.50    0.00    0.50
```

Again, we can reduce the amount of CPU time by heeding some simple rules. One of the reasons for the performance gain is that `2*pi` can be calculated just once [as in `time5()`]; there is no need to calculate it $n = 100000$ times [as in the example in `time4()`].

Do not avoid loops simply for the sake of avoiding loops.

Some time ago, a question was posted to the *R-help* email list asking how to sum a large number of matrices in a list. To simulate this situation, we create a list of 10000 100×100 matrices containing random-normal numbers:

```
R> matrices <- list()
R> for (i in seq_along(matrices))
 +     matrices[[i]] <-
 +         matrix(rnorm(10000), 100, 100)
```

Notice that we do not initialize `matrices` because, as we mentioned, there is no advantage to initializing a list.

One suggestion was to use a loop to sum the matrices, as follows, producing, we claim, simple, straightforward code:

```
R> system.time({
 +     S <- matrix(0, 100, 100)
 +     for (M in matrices)
 +         S <- S + M
 + })
    user  system elapsed
    1.22    0.08    1.30
```

In response, someone else suggested the following 'cleverer' solution, which avoids the loop:

```
R> system.time(S <- apply(array(unlist(matrices),
 +     dim = c(100, 100, 10000)), 1:2, sum))
Error: cannot allocate vector of size 762.9 Mb
```

Not only does this solution fail for a problem of this magnitude on the system on which we tried it (a 32-bit system, hence limited to 2Gb for the process), but it is slower on smaller problems. We invite the reader to redo this problem with 10000 10×10 matrices, for example.

A final note on this problem:

```
R> S <- rowSums(array(unlist(matrices),
 +     dim = c(10, 10, 10000)), dims = 2)
```

is approximately as fast as the loop for the smaller version of the problem but fails on the larger one.

The lesson: Avoid loops to produce clearer and possibly more efficient code, not simply to avoid loops.

Summary

To answer the frequently asked question, 'How can I avoid this loop or make it faster?': Try to use simple vectorized operations; use the family of apply functions if appropriate; initialize objects to full length when using loops; and do not repeat calculations many times if performing them just once is sufficient.

Measure execution time before making changes to code, and only make changes if the efficiency gain really matters. It is better to have readable code that is free of bugs than to waste hours optimizing code to gain a fraction of a second. Sometimes, in fact, a loop will provide a clear and efficient solution to a problem (considering both time and memory use).

Acknowledgment

We would like to thank Bill Venables for his many helpful suggestions.

Bibliography

R. Hyde. The fallacy of premature optimization. *Ubiquity*, 7(24), 2006. URL `http://www.acm.org/ubiquity`.

R Development Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2008a. URL `http://www.R-project.org`.

R Development Core Team. *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, Austria, 2008b. URL `http://www.R-project.org`.

L. Tierney. *proftools: Profile Output Processing Tools for R*, 2007. R package version 0.0-2.

W. Venables. Programmer's Niche. *R News*, 1(1):27–30, 2001. URL http://CRAN.R-project.org/doc/Rnews/.

R. C. Whaley and A. Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, February 2005. URL http://www.cs.utsa.edu/~whaley/papers/spercw04.ps.

H. Wickham. *profr: An alternative display for profiling information*, 2008. URL http://had.co.nz/profr. R package version 0.1.1.

Uwe Ligges
Department of Statistics, Technische Universität Dortmund, Germany
ligges@statistik.tu-dortmund.de

John Fox
Department of Sociology, McMaster University, Hamilton, Ontario, Canada
jfox@mcmaster.ca

Changes in R Version 2.7.0

by the R Core Team

User-visible changes

- The default graphics device in non-interactive use is now `pdf()` rather than `postscript()`. [PDF viewers are now more widely available than PostScript viewers.]

  The default width and height for `pdf()` and `bitmap()` have been changed to 7 (inches) to match the screen devices.

- Most users of the `X11()` device will see a new device that has different fonts, anti-aliasing of lines and fonts and supports semi-transparent colours.

- Considerable efforts have been made to make the default output from graphics devices as similar as possible (and in particular close to that from postscript/pdf). Many devices were misinterpreting 'pointsize' in some way, for example as being in device units (pixels) rather than in points.

- Packages which include graphics devices need to be re-installed for this version of R, with recently updated versions.

New features

- The apse code used by `agrep()` has been updated to version 0.16, with various bug fixes.

  `agrep()` now supports multibyte character sets.

- `any()` and `all()` avoid coercing zero-length arguments (which used a surprising amount of memory) since they cannot affect the answer.

  Coercion of other than integer arguments now gives a warning as this is often a mistake (e.g. writing `all(pr) > 0` instead of `all(pr > 0)`).

- `as.Date()`, `as.POSIXct()` and `as.POSIXlt()` now convert numeric arguments (days or seconds since some epoch) provided the 'origin' argument is specified.

- New function `as.octmode()` to create objects such as file permissions.

- `as.POSIXlt()` is now generic, and it and `as.POSIXct()` gain a '...' argument. The character/factor methods now accept a 'format' argument (analogous to that for as.Date).

- New function `browseVignettes()` lists available vignettes in an HTML browser with links to PDF, Rnw, and R files.

- There are new capabilities "aqua" (for the AQUA GUI and `quartz()` device on Mac OS X) and "cairo" (for cairo-based graphics devices).

- New function `checkNEWS()` in package 'tools' that detects common errors in NEWS file formatting.

- `deparse()` gains a new argument 'nlines' to limit the number of lines of output, and this is used internally to make several functions more efficient.

- `deriv()` now knows the derivatives of `digamma(x)`, `trigamma(x)` and `psigamma(x, deriv)` (wrt to `x`).

- `dir.create()` has a new argument 'mode', used on Unix-alikes (only) to set the permissions on the created directory.

- Where an array is dropped to a length-one vector by `drop()` or [, drop = TRUE], the result now has names if exactly one of the dimensions was named. (This is compatible with S.) Previously there were no names.

- The 'incomparables' argument to `duplicated()`, `unique()` and `match()` is now implemented, and passed to `match()` from `merge()`.

- `dyn.load()` gains a 'DLLpath' argument to specify the path for dependent DLLs: currently only used on Windows.

- The spreadsheet `edit()` methods (and used by `fix()`) for data frames and matrices now warn when classes are discarded.

  When editing a data frame, columns of unknown type (that is not numeric, logical, character or factor) are now converted to character (instead of numeric).

- `file.create()` has a new argument 'showWarnings' (default TRUE) to show an informative warning when creation fails, and `dir.create()` warns under more error conditions.

- New higher-order functions `Find()`, `Negate()` and `Position()`.

- `[dpqr]gamma(*, shape = 0)` now work as limits of 'shape -> 0', corresponding to the point distribution with all mass at 0.

- An informative warning (in addition to the error message) will be given when the basic, extended or perl mode of `grep()`, `strsplit()` and friends fails to compile the pattern.
- More study is done of perl=TRUE patterns in `grep()` and friends when `length(x)` > 10: this should improve performance on long vectors.
- `grep()`, `strsplit()` and friends with fixed=TRUE or perl=TRUE work in UTF-8 and preserve the UTF-8 encoding for UTF-8 inputs where supported.
- `help.search()` now builds the database about 3x times faster.
- `iconv()` now accepts "UTF8" on all platforms (many did, but not e.g. libiconv as used on Windows).
- `identity()` convenience function to be used for programming.
- In addition to warning when 'pkgs' is not found, `install.packages()` now reports if it finds a valid package with only a case mismatch in the name.
- `intToUtf8()` now marks the Encoding of its output.
- The function `is()` now works with S3 inheritance; that is, with objects having multiple strings in the class attribute.
- Extensions to condition number computation for matrices, notably complex ones are provided, both in `kappa()` and the new `rcond()`.
- `list.files()` gains a 'ignore.case' argument, to allow case-insensitive matching on some Windows/MacOS file systems.
- `ls.str()` and `lsf.str()` have slightly changed arguments and defaults such that `ls.str()` no arguments works when debugging.
- Under Unix, `utils::make.packages.html()` can now be used directly to set up linked HTML help pages, optionally without creating the package listing and search database (which can be much faster).
- `new.packages()` now knows about the frontend package gnomeGUI (which does not install into a library).
- `optim(*, control = list(...))` now warns when '...' contains unexpected names, instead of silently ignoring them.
- The options "browser" and "editor" may now be set to functions, just as "pager" already could.
- `packageDescription()` makes use of installed metadata where available (for speed, e.g. in `make.packages.html()`).
- `pairwise.t.test()` and `pairwise.wilcox.test()` now more explicitly allow paired tests. In the former case it is now flagged as an error if both 'paired' and 'pool.SD' are set TRUE (formerly, 'paired' was silently ignored), and one-sided tests are generated according to 'alternative' also if 'pool.SD' is TRUE.
- `paste()` and `file.path()` are now completely internal, for speed. (This speeds up `make.packages.html(packages=FALSE)` severalfold, for example.)
- `paste()` now sets the encoding on the result under some circumstances (see ?paste).
- `predict.loess()` now works when `loess()` was fitted with transformed explanatory variables, e.g, `loess(y ~ log(x)+ log(z))`.
- `print(<data.frame>)`'s new argument 'row.names' allows to suppress printing rownames.
- `print()` and `str()` now also "work" for 'logLik' vectors longer than one.
- Progress-bar functions `txtProgressBar()`, `tkProgressBar()` in package tcltk and `winProgressBar()` (Windows only).
- `readChar()` gains an argument 'useBytes' to allow it to read a fixed number of bytes in an MBCS locale.
- `readNEWS()` has been moved to the tools package.
- `round()` and `signif()` now do internal argument matching if supplied with two arguments and at least one is named.
- New function `showNonASCII()` in package tools to aid detection of non-ASCII characters in .R and .Rd files.
- The `[dpq]signrank()` functions now typically use considerably less memory than previously, thanks to a patch from Ivo Ugrina.
- `spec.ar()` now uses `frequency(x)` when calculating the frequencies of the estimated spectrum, so that for monthly series the frequencies are now per year (as for spec.pgram) rather than per month as before.
- `spline()` gets an 'xout' argument, analogously to `approx()`.

- `sprintf()` now does all the conversions needed in a first pass if `length(fmt) == 1`, and so can be many times faster if called with long vector arguments.

- `[g]sub(useBytes = FALSE)` now sets the encoding on changed elements of the result when working on an element of known encoding. (This was previously done only for perl = TRUE.)

- New function `Sys.chmod()`, a wrapper for 'chmod' on platforms which support it. (On Windows it handles only the read-only bit.)

- New function `Sys.umask()`, a wrapper for 'umask' on platforms which support it.

- New bindings `ttk*()` in package tcltk for the 'themed widgets' of Tk 8.5. The tcltk demos make use of these widgets where available.

- `write.table(d, row.names=FALSE)` is faster when 'd' has millions of rows; in particular for a data frame with automatic row names. (Suggestion from Martin Morgan.)

- The parser limit on string size has been removed.

- If a NEWS file is present in the root of a source package, it is installed (analogously to LICENSE, LICENCE and COPYING).

- Rd conversion to 'example' now quotes aliases which contain spaces.

- The handling of DST on dates outside the range 1902-2037 has been improved. Dates after 2037 are assumed to have the same DST rules as currently predicted for the 2030's (rather than the 1970s), and dates prior to 1902 are assumed to have no DST and the same offset as in 1902 (if known, otherwise as in the 1970s).

- On platforms where we can detect that mktime sets errno (e.g. Solaris and the code used on Windows but not Linux nor Mac OS X), 1969-12-31 23:59:59 GMT is converted from POSIXlt to POSIXct as -1 and not NA.

- The definition of 'whitespace' used by the parser is slightly wider: it includes Unicode space characters on Windows and in UTF-8 locales on machines which use Unicode wide characters.

- The src/extra/intl sources have been updated to those from gettext 0.17.

- New flag –interactive on Unix-alikes forces the session to be interactive (as –ess does on Windows).

- x[<zero-length>] <- NULL is always a no-op: previously type-checking was done on the replacement value and so this failed, whereas we now assume NULL can be promoted to any zero-length vector-like object.

  Other cases of a zero-length index are done more efficiently.

- There is a new option in Rd markup of `\donttest{}` to mark example code that should be run by `example()` but not tested (e.g. because it might fail in some locales).

- The error handler in the parser now reports line numbers for more syntax errors (MBCS and Unicode encoding errors, line length and context stack overflows, and mis-specified argument lists to functions).

- The "MethodsList" objects originally used for method selection are being phased out. New utilities provide simpler alternatives (see ?findMethods), and direct use of the mangled names for the objects is now deprecated.

- Creating new S4 class and method definitions in an environment that could not be identified (as package, namespace or global) previously generated an error. It now results in creating and using an artificial package name from the current date/time, with a warning. See ?getPackageName.

- Unix-alikes now give a warning on startup if locale settings fail. (The Windows port has long done so.)

- Parsing and scanning of numerical constants is now done by R's own C code. This ensures cross-platform consistency, and mitigates the effects of setting LC_NUMERIC (within base R it only applies to output – packages may differ).

  The format accepted is more general than before and includes binary exponents in hexadecimal constants: see ?NumericConstants for details.

- Dependence specifications for R or packages in the Depends field in a DESCRIPTION file can now make use of operators < > == and != (in addition to <= and >=): such packages will not be installable nor loadable in R < 2.7.0.

  There can be multiple mentions of R or a package in the Depends field in a DESCRIPTION file: only the first mention will be used in R < 2.7.0.

  GRAPHICS CHANGES

- The default graphics devices in interactive and non-interactive sessions are

now configurable via environment variables R_INTERACTIVE_DEVICE and R_DEFAULT_DEVICE respectively.

- New function `dev.new()` to launch a new copy of the default graphics device (and taking care if it is "pdf" or "postscript" not to trample on the file of an already running copy).
- `dev.copy2eps()` uses `dev.displaylist()` to detect screen devices, rather than list them in the function.
- New function `dev.copy2pdf()`, the analogue of `dev.copy2eps()`.
- `dev.interactive()` no longer treats a graphics device as interactive if it has a display list (but devices can still register themselves on the list of interactive devices).
- The `X11()` and `windows()` graphics devices have a new argument 'title' to set the window title.
- `X11()` now has the defaults for all of its arguments set by the new function `X11.options()`, inter alia replacing options "gamma", "colortype" and "X11fonts".
- `ps.options()` now warns on unused option 'append'.

  `xfig()` no longer takes default arguments from `ps.options()`. (This was not documented prior to 2.6.1 patched.)

  `pdf()` now takes defaults from the new function `pdf.options()` rather that from `ps.options()` (and the latter was not documented prior to 2.6.1 patched).

  The defaults for all arguments other than 'file' in `postscript()` and `pdf()` can now be set by `ps.options()` or `pdf.options()`
- New functions `setEPS()` and `setPS()` as wrappers to `ps.options()` to set appropriate defaults for figures for inclusion in other documents and for spooling to a printer respectively.
- The meaning of numeric 'pch' has been extended where MBCSes are supported. Now negative integer values indicate Unicode points, integer values in 32-127 represent ASCII characters, and 128-255 are valid only in single-byte locales. (Previously what happened with negative pch values was undocumented: they were replaced by the current setting of `par("pch")`.)
- Graphics devices can say if they can rotate text well (e.g. `postscript()` and `pdf()` can) and if so the device's native text becomes the default for contour labels rather than using Hershey fonts.
- The setting of the line spacing (`par("cra")[2]`) on the `X11()` and `windows()` devices is now comparable with `postscript()` etc, and roughly 20% smaller than before (it used to depend on the locale for X11). (So is the `pictex()` device, now 20% larger.) This affects the margin size in plots, and should result in better-looking plots.
- There is a per-device setting for whether new frames need confirmation. This is controlled by either `par("ask")` or `grid.prompt()` and affects all subsequent plots on the device using base or grid graphics.
- There is a new version of the `X11()` device based on cairo graphics which is selected by type "cairo" or "nbcairo", and is available on machines with cairo installed and preferably pango (which most machines with gtk+ >= 2.8 will have). This version supports translucent colours and normally does a better job of font selection so it has been possible to display (e.g.) English, Polish, Russian and Japanese text on a single `X11()` window. It is the default where available.

  There is a companion function, `savePlot()`, to save the current plot to a PNG file.

  On Unix-alikes, devices `jpeg()` and `png()` also accept type = "cairo", and with that option do not need a running X server. The meaning of `capabilities("jpeg")` and `capabilities("png")` has changed to reflect this. On MacOS X, there is a further type = "quartz". The default type is selected by the new option "bitmapType", and is "quartz" or "cairo" where available.

  Where cairo 1.2 or later is supported, there is a `svg()` device to write SVG files, and `cairo_pdf()` and `cairo_ps()` devices to write (possibly bitmap) PDF and postscript files via cairo.

  Some features require cairo >= 1.2, and some which are nominally supported under 1.2 seem to need 1.4 to work well.
- There are new `bmp()` and `tiff()` devices.
- New function `devSize()` to report the size of the current graphics device surface (in inches or device units). This gives the same information as `par("din")`, but independent of the graphics subsystem.
- New base graphics function `clip()` to set the clipping region (in user coordinates).

- New functions `grconvertX()` and `grconvertY()` to convert between coordinate systems in base graphics.
- `identify()` recycles its 'labels' argument if necessary.
- `stripchart()` is now a generic function, with default and formula methods defined. Additional graphics parameters may be included in the call. Formula handling is now similar to `boxplot()`.
- `strwidth()` and `strheight()` gain 'font' and 'vfont' arguments and accept in-line pars such as 'family' in the same way as `text()` does. (Longstanding wish of PR#776)
- `example(ask=TRUE)` now applies to grid graphics (e.g. from lattice) as well as to base graphics.
- Option "device.ask.default" replaces "par.ask.-default" now it applies also to `grid.prompt()`.
- `plot.formula()` only prompts between plots for interactive devices (it used to prompt for all devices).
- When `plot.default()` is called with y=NULL it now calls `Axis()` with the 'y' it constructs rather than use the default axis.

Deprecated & defunct

- In package installation, SaveImage: yes is defunct and lazyloading is attempted instead.
- `$` on an atomic vector or S4 object is now defunct.
- Partial matching in [[is now only performed if explicitly requested (by exact=FALSE or exact=NA).
- Command-line completion has been moved from package 'rcompgen' to package 'utils': the former no longer exists as a separate package in the R distribution.
- The S4 pseudo-classes "single" and double have been removed. (The S4 class for a REALSXP is "numeric": for back-compatibility `as(x, "double")` coerces to "numeric".)
- `gpar(gamma=)` in the grid package is now defunct.
- Several S4 class definition utilities, `get*()`, have been said to be deprecated since R 1.8.0; these are now formally deprecated. Ditto for `removeMethodsObject()`.
- Use of the graphics headers Rgraphics.h and Rdevices.h is deprecated, and these will be unavailable in R 2.8.0. (They are hardly used except in graphics devices, for which there is an updated API in this version of R.)
- `options("par.ask.default")` is deprecated in favour of "device.ask.default".
- The 'device-independent' family "symbol" is deprecated as it was highly locale- and device-dependent (it only did something useful in single-byte locales on most devices) and font=5 (base) or fontface=5 (grid) did the job it was intended to do more reliably.
- `gammaCody()` is now formally deprecated.
- Two low-level functions using MethodsList metadata objects (`mlistMetaName()` and `getAllMethods()`) are deprecated.
- Setting `par(gamma=)` is now deprecated, and the `windows()` device (the only known example) no longer allows it.
- The C macro 'allocString' will be removed in 2.8.0 – use 'mkChar', or 'allocVector' directly if really necessary.

Installation

- Tcl/Tk >= 8.3 (released in 2000) is now required to build package tcltk.
- configure first tries TCL_INCLUDE_SPEC and TK_INCLUDE_SPEC when looking for Tcl/Tk headers. (The existing scheme did not work for the ActiveTcl package on Mac OS X.)
- The Windows build only supports Windows 2000 or later (XP, Vista, Server 2003 and Server 2008).
- New option –enable-R-static-lib installs libR.a which can be linked to a front-end via 'R CMD config –ldflags'. The tests/Embedding examples now work with a static R library.
- Netscape (which was discontinued in Feb 2008) is no longer considered when selecting a browser.
- xdg-open (the freedesktop.org interface to kfmclient/gnome-open/...) is considered as a possible browser, after real browsers such as firefox, mozilla and opera.
- The search for tclConfig.sh and tkConfig.sh now only looks in directories with names containing `$(LIBnn)` in the hope of finding the version for the appropriate architecture (e.g. x86_64 or i386).

- libtool has been updated to version 2.2.
- Use of –with-system-zlib, –with-system-bzlib or –with-system-pcre now requires version >= 1.2.3, 1.0.5, 7.6 respectively, for security.

Utilities

- Rdconv now removes empty sections including alias and keyword entries, with a note.
- Keyword entries are no longer mandatory in Rd files.
- R CMD INSTALL now also installs tangled versions of all vignettes.
- R CMD check now warns if spaces or non-ASCII characters are used in file paths, since these are not in general portable.
- R CMD check (via massage-examples.pl) now checks all examples with a 7 inch square device region on A4 paper, for locale-independence and to be similar to viewing examples on an on-screen device.

  If a package declares an encoding in the DESCRIPTION file, the examples are assumed to be in that encoding when running the tests. (This avoids errors in running latin1 examples in a UTF-8 locale.)
- R CMD check uses pdflatex (if available) to check the typeset version of the manual, producing PDF rather than DVI. (This is a better check since the package reference manuals on CRAN are in PDF.)
- R CMD Rd2dvi gains a –encoding argument to be passed to R CMD Rdconv, to set the default encoding for conversions. If this is not supplied and the files are package sources and the DESCRIPTION file contains an Encoding field, that is used for the default encoding.
- `available.packages()` (and hence `install.packages()` etc.) now supports subdirectories in a repository, and `tools::write_PACKAGES()` can now produce PACKAGES files including subdirectories.
- The default for 'stylepath' in Sweave's (default) RweaveLatex driver can be set by the environment variable SWEAVE_STYLEPATH_DEFAULT: see ?RweaveLatex.

C-level facilities

- Both the Unix and Windows interfaces for embedding now make use of 'const char *' declarations where appropriate.
- `Rprintf()` and `REprintf()` now use 'const char *' for their format argument – this should reduce warnings when called from C++.
- There is a new description of the interface for graphics devices in the 'R Internals' manual, and several new entry points. The API has been updated to version R_GE_version = 5, and graphics devices will need to be updated accordingly.
- Graphics devices can now select to be sent text in UTF-8, even if the current locale is not UTF-8 (and so enable text entered in UTF-8 to be plotted). This is used by `postscript()`, `pdf()` and the `windows()` family of devices, as well as the new cairo-based devices.
- More Lapack routines are available (and declared in R_Ext/Lapack.h), notably for (reciprocal) condition number estimation of complex matrices.
- Experimental utility R_has_slot supplementing R_do_slot.
- There is a new public interface to the encoding info stored on CHARSXPs, getCharCE and mkCharCE using the enumeration type cetype_t.
- A new header 'R_ext/Visibility.h' contains some definitions for controlling the visibility of entry points, and how to control visibility is now documented in 'Writing R Extensions'.

Bug fixes

- `pt(x, df)` is now even more accurate in some cases (e.g. 12 instead of 8 significant digits), when $x^2 << df$, thanks to a remark from Ian Smith, related to PR#9945.
- `co[rv](use = "complete.obs")` now always gives an error if there are no complete cases: they used to give NA if method = "pearson" but an error for the other two methods. (Note that this is pretty arbitrary, but zero-length vectors always give an error so it is at least consistent.)

  `cor(use="pair")` used to give diagonal 1 even if the variable was completely missing for the rank methods but NA for the Pearson method: it now gives NA in all cases.

  `cor(use="pair")` for the rank methods gave a matrix result with dimensions > 0 even if one of the inputs had 0 columns.
- Supplying edit.row.names = TRUE when editing a matrix without row names is now an error and not a segfault. (PR#10500)

- The error handler in the parser reported unexpected & as && and | as ||.
- `ps.options(reset = TRUE)` had not reset for a long time.
- `paste()` and `file.path()` no longer allow NA_character_ for their 'sep' and 'collapse' arguments.
- `by()` failed for 1-column matrices and dataframes. (PR#10506) However, to preserve the old behaviour, the default method when operating on a vector still passes subsets of the vector to FUN, and this is now documented.
- Better behaviour of `str.default()` for non-default 'strict.width' (it was calling `str()` rather than `str.default()` internally); also, more useful handling of `options("str")`.
- `wilcox.test(exact=FALSE, conf.int=TRUE)` could fail in some extreme two-sample problems. (Reported by Wolfgang Huber.)
- `par(pch=)` would accept a multi-byte string but only use the first byte. This would lead to incorrect results in an MBCS locale if a non-ASCII character was supplied.
- There are some checks for valid C-style formats in, e.g. `png(filename=)`. (PR#10571)
- `vector()` was misinterpreting some double 'length' values, e.g, NaN and NA_real_ were interpreted as zero. Also, invalid types of 'length' were interpreted as -1 and hence reported as negative. (length<- shared the code and hence the same misinterpretations.)
- A basic class "S4" was added to correspond to the "S4" object type, so that objects with this type will print, etc. The class is VIRTUAL, since all actual S4 objects must have a real class.
- Classes with no slots that contain only VIRTUAL classes are now VIRTUAL, as was intended but confused by having an empty S4 object as prototype. ## backed out temporarily ##
- `format.AsIs()` discarded dimnames, causing dataframes with matrix variables to be printed without using the column names, unlike what happens in S-PLUS (Tim Hesterberg, PR#10730).
- `xspline()` and `grid::grid.xspline()` work in device coordinates and now correct for anisotropy in the device coordinate system.
- `grid.locator()` now indicates to the graphics device that it is is in 'graphics input' mode (as `locator()` and `identify()` always have). This means that devices can now indicate the 'graphics input' mode by e.g. a change of cursor.
- Locales without encoding specification and non-UTF-8 locales now work properly on Mac OS X. Note that locales without encoding specification always use UTF-8 encoding in Mac OS X (except for specials "POSIX" and "C") - this is different from other operating systems.
- `iconv()` now correctly handles to="" and from="" on Mac OS X.
- In `diag()`'s argument list, drop the explicit default (' = n') for 'ncol' which is ugly when making `diag()` generic.
- S4 classes with the same name from different packages were not recognized because of a bug in caching the new definition.
- `jpeg()` and `png()` no longer maintain a display list, as they are not interactive devices.
- Using `attr(x, "names") <-` value (instead of the correct `names<-`) with 'value' a pairlist (instead of the correct character vector) worked incorrectly. (PR#10807)
- Using `[<-` to add a column to a data frame dropped other attributes whereas `[[<-` and `$<-` did not: now all preserve attributes. (PR#10873)
- File access functions such as `file.exists()`, `file.info()`, `dirname()` and `unlink()` now treat an NA filename as a non-existent file and not the file "NA".
- `r<foo>()`, the random number generators, are now more consistent in warning when NA's (specifically NaN's) are generated.
- `rnorm(n, mu = Inf)` now returns `rep(Inf, n)` instead of NaN; similar changes are applied to `rlnorm()`, `rexp()`, etc.
- `[l]choose()` now warns when rounding non-integer 'k' instead of doing so silently. (May help confused users such as PR#10766.)
- `gamma()` was warning incorrectly for most negative values as being too near a negative integer. This also affected other functions making use of its C-level implementation.
- `dumpMethod()` and `dumpMethods()` now work again.
- `package.skeleton()` now also works for code_files with only metadata (e.g. S4 setClass) definitions; it handles S4 classes and methods, producing documentation and NAMESPACE exports if requested.

- Some methods package utilities (`implicitGeneric()`, `makeGeneric()`) will be more robust in dealing with primitive functions (not a useful idea to call them with primitives, though).
- Making a MethodsList from a function with no methods table will return an empty list, rather than cause an error (questionably a bug, but caused some obscure failures).
- `setAs()` now catches 2 arguments in the method definition, if they do not match the arguments of `coerce()`.
- S4 methods with missing arguments in the definition are handled correctly when non-signature arguments exist, and check for conflicting local names in the method definition.
- `qgamma()` and `qchisq()` could be inaccurate for small p, e.g. `qgamma(1.2e-10, shape = 19)` was 2.52 rather than 2.73.
- `dbeta(.., ncp)` is now more accurate for large ncp, and typically no longer underflows for give.log = TRUE.
- `coerce()` is now a proper S4 object and so prints correctly.
- @ now checks it is being applied to an S4 object, and if not gives a warning (which will become an error in 2.8.0).
- `dump()` and friends now warn that all S4 objects (even those based on vectors) are not `source()`able, with a stronger wording.
- `read.dcf(all = TRUE)` was leaking connections.
- `scan()` with a non-default separator could skip nul bytes, including those entered as code 0 with allowEscapes=TRUE. This was different from the default separator.
- `determinant(matrix(,0,0))` now returns a correct "det" result; also value 1 or 0 depending on 'logarithm', rather than `numeric(0)`.
- Name space 'grDevices' was not unloading its DLL when the name space was unloaded.
- `getNativeSymbolInfo()` was unaware of non-registered Fortran names, because one of the C support routines ignored them.
- `load()` again reads correctly character strings with embedded nuls. (This was broken in 2.6.x, but worked in earlier versions.)

Changes on CRAN

by Kurt Hornik

CRAN package web

The CRAN package web area has substantially been reorganized and enhanced. Most importantly, packages now have persistent package URLs of the form

`http://CRAN.R-project.org/package=`*foo*

which is also the recommended package URL for citations. (The `package=`*foo* redirections also work for most official CRAN mirrors.) The corresponding package web page now has its package dependency information hyperlinked. It also points to a package check page with check results and timings, and to an archive directory with the sources of older versions of the package (if applicable), which are conveniently gathered into an 'Archive/*foo*' subdirectory of the CRAN 'src/contrib' area.

CRAN package checking

The CRAN Linux continuing ("daily") check processes now fully check packages with dependencies on packages in Bioconductor and Omegahat. All check flavors now give timings for installing and checking installed packages. The check results are available in overall summaries sorted by either package name or maintainer name, and in individual package check summary pages.

New contributed packages

BAYSTAR Bayesian analysis of Threshold autoregressive model (BAYSTAR). By Cathy W. S. Chen, Edward M.H. Lin, F.C. Liu, and Richard Gerlach.

BB Barzilai-Borwein Spectral Methods for solving nonlinear systems of equations, and for optimizing nonlinear objective functions subject to simple constraints. By Ravi Varadhan.

BPHO Bayesian Prediction with High-Order interactions. This software can be used in two situations. The first is to predict the next outcome based on the previous states of a discrete sequence. The second is to classify a discrete response based on a number of discrete covariates. In both situations, we use Bayesian logistic regression models that consider the high-order interactions. The models are trained with slice sampling method, a variant of Markov chain Monte Carlo. The time arising from using high-order interactions is reduced greatly by our compression technique that represents a group of original parameters as a single one in MCMC step. By Longhai Li.

Bchron Create chronologies based on radiocarbon and non-radiocarbon dated depths, following the work of Parnell and Haslett (2007, JRSSC). The package runs MCMC, predictions and plots for radiocarbon (and non radiocarbon) dated sediment cores. By Andrew Parnell.

BootPR Bootstrap Prediction Intervals and Bias-Corrected Forecasting for auto-regressive time series. By Jae H. Kim.

COZIGAM COnstrained Zero-Inflated Generalized Additive Model (COZIGAM) fitting with associated model plotting and prediction. By Hai Liu and Kung-Sik Chan.

CPE Concordance Probability Estimates in survival analysis. By Qianxing Mo, Mithat Gonen and Glenn Heller.

ChainLadder Mack- and Munich-chain-ladder methods for insurance claims reserving. By Markus Gesmann.

CombMSC Combined Model Selection Criteria: functions for computing optimal convex combinations of model selection criteria based on ranks, along with utility functions for constructing model lists, MSCs, and priors on model lists. By Andrew K. Smith.

Containers Object-oriented data structures for R: stack, queue, deque, max-heap, min-heap, binary search tree, and splay tree. By John Hughes.

CoxBoost Cox survival models by likelihood based boosting. By Harald Binder.

DEA Data Envelopment Analysis, performing some basic models in both multiplier and envelopment form. By Zuleyka Diaz-Martinez and Jose Fernandez-Menendez.

DierckxSpline R companion to "Curve and Surface Fitting with Splines", providing a wrapper to the FITPACK routines written by Paul Dierckx. The original Fortran is available from `http://www.netlib.org/dierckx`. By Sundar Dorai-Raj.

EMC Evolutionary Monte Carlo (EMC) algorithm. Random walk Metropolis, Metropolis Hasting, parallel tempering, evolutionary Monte Carlo, temperature ladder construction and placement. By Gopi Goswami.

EMCC Evolutionary Monte Carlo (EMC) methods for clustering, temperature ladder construction and placement. By Gopi Goswami.

EMD Empirical Mode Decomposition and Hilbert spectral analysis. By Donghoh Kim and Hee-Seok Oh.

ETC Tests and simultaneous confidence intervals for equivalence to control. The package allows selecting those treatments of a one-way layout being equivalent to a control. Bonferroni adjusted "two one-sided *t*-tests" (TOST) and related simultaneous confidence intervals are given for both differences or ratios of means of normally distributed data. For the case of equal variances and balanced sample sizes for the treatment groups, the single-step procedure of Bofinger and Bofinger (1995) can be chosen. For non-normal data, the Wilcoxon test is applied. By Mario Hasler.

EffectiveDose Estimates the Effective Dose level for quantal bioassay data by nonparametric techniques and gives a bootstrap confidence interval. By Regine Scheder.

FAiR Factor Analysis in R. This package estimates factor analysis models using a genetic algorithm, which opens up a number of new ways to pursue old ideas, such as those discussed by Allen Yates in his 1987 book "Multivariate Exploratory Data Analysis". The major sources of value added in this package are new ways to transform factors in exploratory factor analysis, and perhaps more importantly, a new estimator for the factor analysis model called semi-exploratory factor analysis. By Ben Goodrich.

FinTS R companion to Tsay (2005), "Analysis of Financial Time Series, 2nd ed." (Wiley). Includes data sets, functions and script files required to work some of the examples. Version 0.2-x includes R objects for all data files used in the text and script files to recreate most of the analyses in chapters 1 and 2 plus parts of chapters 3 and 11. By Spencer Graves.

FitAR Subset AR Model fitting. Complete functions are given for model identification, estimation and diagnostic checking for AR and subset AR models. Two types of subset AR models are supported. One family of subset AR models, denoted by ARp, is formed by taking subsets of the original AR coefficients and in the other, denoted by ARz, subsets of the partial autocorrelations are used. The main advantage of the ARz model is its applicability to very large order models. By A.I. McLeod and Ying Zhang.

FrF2 Analyzing Fractional Factorial designs with 2-level factors. The package is meant for completely aliased designs only, i.e., e.g. not for analyzing Plackett-Burman designs with interactions. Enables convenient main effects and interaction plots for all factors simultaneously and offers a cube plot for looking at the simultaneous effects of three factors. An enhanced `DanielPlot` function (modified from **BsMD**) is provided. Furthermore, the alias structure for Fractional Factorial 2-level designs is output in a more readable format than with the built-in function alias. By Ulrike Groemping.

FunNet Functional analysis of gene co-expression networks from microarray expression data. The analytic model implemented in this package involves two abstraction layers: transcriptional and functional (biological roles). A functional profiling technique using Gene Ontology & KEGG annotations is applied to extract a list of relevant biological themes from microarray expression profiling data. Afterwards, multiple-instance representations are built to relate significant themes to their transcriptional instances (i.e., the two layers of the model). An adapted non-linear dynamical system model is used to quantify the proximity of relevant genomic themes based on the similarity of the expression profiles of their gene instances. Eventually an unsupervised multiple-instance clustering procedure, relying on the two abstraction layers, is used to identify the structure of the co-expression network composed from modules of functionally related transcripts. Functional and transcriptional maps of the co-expression network are provided separately together with detailed information on the network centrality of related transcripts and genomic themes. By Corneliu Henegar.

GEOmap Routines for making map projections (forward and inverse), topographic maps, perspective plots, geological maps, geological map symbols, geological databases, interactive plotting and selection of focus regions. By Jonathan M. Lees.

IBrokers R API to Interactive Brokers Trader Workstation. By Jeffrey A. Ryan.

ISA Insieme di funzioni di supporto al volume "INTRODUZIONE ALLA STATISTICA APPLICATA con esempi in R", Federico M. Stefanini, PEARSON Education Milano, 2007. By Fabio Frascati and Federico M. Stefanini.

ISOcodes ISO language, territory, currency, script and character codes. Provides ISO 639 language codes, ISO 3166 territory codes, ISO 4217 currency codes, ISO 15924 script codes, and the ISO 8859 and ISO 10646 character codes as well

as the Unicode data table. By Christian Buchta and Kurt Hornik.

Iso Functions to perform isotonic regression. Does linear order and unimodal order isotonic regression. By Rolf Turner.

JM Shared parameter models for the joint modeling of longitudinal and time-to-event data. By Dimitris Rizopoulos.

MCPAN Multiple contrast tests and simultaneous confidence intervals based on normal approximation. With implementations for binomial proportions in a $2 \times k$ setting (risk difference and odds ratio), poly-3-adjusted tumor rates, and multiple comparisons of biodiversity indices. Approximative power calculation for multiple contrast tests of binomial proportions. By Frank Schaarschmidt, Daniel Gerhard, and Martin Sill.

MCPMod Design and analysis of dose-finding studies. Implements a methodology for dose-response analysis that combines aspects of multiple comparison procedures and modeling approaches (Bretz, Pinheiro and Branson, 2005, Biometrics 61, 738–748). The package provides tools for the analysis of dose finding trials as well as a variety of tools necessary to plan a trial to be conducted with the MCPMod methodology. By Bjoern Bornkamp, Jose Pinheiro, and Frank Bretz.

MultEq Tests and confidence intervals for comparing two treatments when there is more than one primary response variable (endpoint) given. The step-up procedure of Quan et al. (2001) is both applied for differences and extended to ratios of means of normally distributed data. A related single-step procedure is also available. By Mario Hasler.

PARccs Estimation of partial attributable risks (PAR) from case-control data, with corresponding percentile or BCa confidence intervals. By Christiane Raemsch.

PASWR Data and functions for the book "Probability and Statistics with R" by M. D. Ugarte, A. F. Militino, and A. T. Arnholt (2008, Chapman & Hall/CRC). By Alan T. Arnholt.

Peaks Spectrum manipulation: background estimation, Markov smoothing, deconvolution and peaks search functions. Ported from ROOT/TSpectrum class. By Miroslav Morhac.

PwrGSD Tools to compute power in a group sequential design. SimPwrGSD C-kernel is a simulation routine that is similar in spirit to 'dssp2.f' by Gu and Lai, but with major improvements. AsyPwrGSD has exactly the same range of application as SimPwrGSD but uses asymptotic methods and runs *much* faster. By Grant Izmirlian.

QuantPsyc Quantitative psychology tools. Contains functions useful for data screening, testing moderation, mediation and estimating power. By Thomas D. Fletcher.

R.methodsS3 Methods that simplify the setup of S3 generic functions and S3 methods. Major effort has been made in making definition of methods as simple as possible with a minimum of maintenance for package developers. For example, generic functions are created automatically, if missing, and name conflict are automatically solved, if possible. The method `setMethodS3()` is a good start for those who in the future want to migrate to S4. This is a cross-platform package implemented in pure R and is generating standard S3 methods. By Henrik Bengtsson.

RExcelInstaller Integration of R and Excel (use R in Excel, read/write XLS files). RExcel, an add-in for MS Excel on MS Windows, allows to transfer data between R and Excel, writing VBA macros using R as a library for Excel, and calling R functions as worksheet function in Excel. RExcel integrates nicely with R Commander (**Rcmdr**). This R package installs the Excel add-in for Excel versions from 2000 to 2007. It only works on MS Windows. By Erich Neuwirth, with contributions by Richard Heiberger and Jurgen Volkering.

RFreak An R interface to a modified version of the Free Evolutionary Algorithm Kit FrEAK (`http://sourceforge.net/projects/freak427/`), a toolkit written in Java to design and analyze evolutionary algorithms. Both the R interface an extended version of FrEAK are contained in the RFreak package. By Robin Nunkesser.

RSEIS Tools for seismic time series analysis via spectrum analysis, wavelet transforms, particle motion, and hodograms. By Jonathan M. Lees.

RSeqMeth Package for analysis of Sequenom EpiTYPER Data. By Aaron Statham.

RTOMO Visualization for seismic tomography. Plots tomographic images, and allows one to interact and query three-dimensional tomographic models. Vertical cross-sectional cuts can be extracted by mouse click. Geographic information can be added easily. By Jonathan M. Lees.

RankAggreg Performs aggregation of ordered lists based on the ranks using three different algorithms: Cross-Entropy Monte Carlo algorithm,

Genetic algorithm, and a brute force algorithm (for small problems). By Vasyl Pihur, Somnath Datta, and Susmita Datta.

RcmdrPlugin.Export Graphically export objects to LaTeX or HTML. This package provides facilities to graphically export **Rcmdr** output to LaTeX or HTML code. Essentially, at the moment, the plug-in is a graphical front-end to `xtable()`. It is intended to (1) facilitate exporting **Rcmdr** output to formats other than ASCII text and (2) provide R novices with an easy to use, easy to access reference on exporting R objects to formats suited for printed output. By Liviu Andronic.

RcmdrPlugin.IPSUR Accompanies "Introduction to Probability and Statistics Using R" by G. Andy Chang and G. Jay Kerns (in progress). Contributes functions unique to the book as well as specific configuration and selected functionality to the R Commander by John Fox. By G. Jay Kerns, with contributions by Theophilius Boye and Tyler Drombosky, adapted from the work of John Fox et al.

RcmdrPlugin.epack Rcmdr plugin for time series. By Erin Hodgess.

Rcplex R interface to CPLEX solvers for linear, quadratic, and (linear and quadratic) mixed integer programs. A working installation of CPLEX is required. Support for Windows platforms is currently not available. By Hector Corrada Bravo.

Rglpk R interface to the GNU Linear Programing Kit (GLPK). GLPK is open source software for solving large-scale linear programming (LP), mixed integer linear programming (MILP) and other related problems. By Kurt Hornik and Stefan Theussl.

Rsymphony An R interface to the SYMPHONY MILP solver (version 5.1.7). By Reinhard Harter, Kurt Hornik and Stefan Theussl.

SMC Sequential Monte Carlo (SMC) Algorithm, and functions for particle filtering and auxiliary particle filtering. By Gopi Goswami.

SyNet Inference and analysis of sympatry networks. Infers sympatry matrices from distributional data and analyzes them in order to identify groups of species cohesively sympatric. By Daniel A. Dos Santos.

TSA Functions and data sets detailed in the book "Time Series Analysis with Applications in R (second edition)" by Jonathan Cryer and Kung-Sik Chan. By Kung-Sik Chan.

TSHRC Two-stage procedure for comparing hazard rate functions which may or may not cross each other. By Jun Sheng, Peihua Qiu, and Charles J. Geyer.

VIM Visualization and Imputation of Missing values. Can be used for exploring the data and the structure of the missing values. Depending on this structure, the tool can be helpful for identifying the mechanism generating the missings. A graphical user interface allows an easy handling of the implemented plot methods. By Matthias Templ.

WINRPACK Reads in WIN pickfile and waveform file, prepares data for **RSEIS**. By Jonathan M. Lees.

XReg Implements extreme regression estimation as described in LeBlanc, Moon and Kooperberg (2006, Biostatistics 7, 71–84). By Michael LeBlanc.

adk Anderson-Darling *K*-sample test and combinations of such tests. By Fritz Scholz.

anacor Simple and canonical correspondence analysis. Performs simple correspondence analysis (CA) on a two-way frequency table (with missings) by means of SVD. Different scaling methods (standard, centroid, Benzecri, Goodman) as well as various plots including confidence ellipsoids are provided. By Jan de Leeuw and Patrick Mair.

anapuce Functions for normalization, differential analysis of microarray data and others functions implementing recent methods developed by the Statistic and Genome Team from UMR 518 AgroParisTech/INRA Appl. Math. Comput. Sc. By J. Aubert.

backfitRichards Computation and plotting of backfitted independent values of Richards curves. By Jens Henrik Badsberg.

bentcableAR Bent-Cable regression for independent data or auto-regressive time series. The bent cable (linear-quadratic-linear) generalizes the broken stick (linear-linear), which is also handled by this package. By Grace Chiu.

biclust BiCluster Algorithms. The main function `biclust()` provides several algorithms to find biclusters in two-dimensional data: Cheng and Church, Spectral, Plaid Model, Xmotifs and Bimax. In addition, the package provides methods for data preprocessing (normalization and discretization), visualization, and validation of bicluster solutions. By Sebastian Kaiser, Rodrigo Santamaria, Roberto Theron, Luis Quintales and Friedrich Leisch.

bifactorial Global and multiple inferences for given bi- and trifactorial clinical trial designs using bootstrap methods and a classical approach. By Peter Frommolt.

bipartite Visualizes bipartite networks and calculates some ecological indices. By Carsten F. Dormann and Bernd Gruber, with additional code from Jochen Fruend, based on the C-code developed by Nils Bluethgen.

birch Dealing with very large data sets using BIRCH. Provides an implementation of the algorithms described in Zhang et al. (1997), and provides functions for creating CF-trees, along with algorithms for dealing with some combinatorial problems, such as `covMcd` and `ltsReg`. It is very well suited for dealing with very large data sets, and does not require that the data can fit in physical memory. By Justin Harrington and Matias Salibian-Barrera.

brglm Bias-reduction in binomial-response GLMs. Fit binomial-response GLMs using either a modified-score approach to bias-reduction or maximum penalized likelihood where penalization is by Jeffreys invariant prior. Fitting takes place by iteratively fitting a local GLM on a pseudo-data representation. The interface is essentially the same as `glm`. More flexibility is provided by the fact that custom pseudo-data representations can be specified and used for model fitting. Functions are provided for the construction of confidence intervals for the bias-reduced estimates. By Ioannis Kosmidis.

bspec Bayesian inference on the (discrete) power spectrum of time series. By Christian Roever.

bvls An R interface to the Stark-Parker algorithm for bounded-variable least squares. By Katharine M. Mullen.

candisc Functions for computing and graphing canonical discriminant analyses. By Michael Friendly and John Fox.

cheb Discrete linear Chebyshev approximation. By Jan de Leeuw.

chemometrics An R companion to the book "Introduction to Multivariate Statistical Analysis in Chemometrics" by K. Varmuza and P. Filzmoser (CRC Press). By P. Filzmoser and K. Varmuza.

cir Nonparametric estimation of monotone functions via isotonic regression and centered isotonic regression. Provides the well-known isotonic regression (IR) algorithm and an improvement called Centered Isotonic Regression (CIR) for the case the true function is known to be smooth and strictly monotone. Also features percentile estimation for dose-response experiments (e.g., ED50 estimation of a medication) using CIR. By Assaf P. Oron.

compHclust Performs the complementary hierarchical clustering procedure and returns X' (the expected residual matrix), and a vector of the relative gene importance. By Gen Nowak and Robert Tibshirani.

contfrac Various utilities for evaluating continued fractions. By Robin K. S. Hankin.

corrperm Three permutation tests of correlation useful when there are repeated measurements. By Douglas M. Potter.

crank Functions for completing and recalculating rankings. By Jim Lemon.

degreenet Likelihood-based inference for skewed count distributions used in network modeling. Part of the "statnet" suite of packages for network analysis. By Mark S. Handcock.

depmixS4 Fit latent (hidden) Markov models on mixed categorical and continuous (time series) data, otherwise known as dependent mixture models. By Ingmar Visser and Maarten Speekenbrink.

diagram Visualization of simple graphs (networks) based on a transition matrix, utilities to plot flow diagrams and visualize webs, and more. Support for the book "A guide to ecological modelling" by Karline Soetaert and Peter Herman (in preparation). By Karline Soetaert.

dynamicTreeCut Methods for detection of clusters in hierarchical clustering dendrograms. By Peter Langfelder and Bin Zhang, with contributions from Steve Horvath.

emu Provides an interface to the Emu speech database system and many special purpose functions for display and analysis of speech data. By Jonathan Harrington and others.

epiR Functions for analyzing epidemiological data. Contains functions for directly and indirectly adjusting measures of disease frequency, quantifying measures of association on the basis of single or multiple strata of count data presented in a contingency table, and computing confidence intervals around incidence risk and incidence rate estimates. Miscellaneous functions for use in meta-analysis, diagnostic test interpretation, and sample size calculations. By Mark Stevenson with contributions from Telmo Nunes, Javier Sanchez, and Ron Thornton.

ergm An integrated set of tools to analyze and simulate networks based on exponential-family random graph models (ERGM). Part of the "statnet" suite of packages for network analysis. By Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris.

fpca A geometric approach to MLE for functional principal components. By Jie Peng and Debashis Paul.

gRain Probability propagation in graphical independence networks, also known as probabilistic expert systems (which includes Bayesian networks as a special case). By Søren Højsgaard.

geozoo Zoo of geometric objects, allowing for display in GGobi through the use of **rggobi**. By Barret Scloerke, with contributions from Dianne Cook and Hadley Wickham.

getopt C-like getopt behavior. Use this with Rscript to write "`#!`" shebang scripts that accept short and long flags/options. By Allen Day.

gibbs.met Naive Gibbs sampling with Metropolis steps. Provides two generic functions for performing Markov chain sampling in a naive way for a user-defined target distribution which involves only continuous variables. `gibbs_met()` performs Gibbs sampling with each 1-dimensional distribution sampled with Metropolis update using Gaussian proposal distribution centered at the previous state. `met_gaussian` updates the whole state with Metropolis method using independent Gaussian proposal distribution centered at the previous state. The sampling is carried out without considering any special tricks for improving efficiency. This package is aimed at only routine applications of MCMC in moderate-dimensional problems. By Longhai Li.

gmaps Extends the functionality of the **maps** package for the **grid** graphics system. This enables more advanced plots and more functionality. It also makes use of the grid structure to fix problems encountered with the traditional graphics system, such as resizing of graphs. By Andrew Redd.

goalprog Functions to solve weighted and lexicographical goal programming problems as specified by Lee (1972) and Ignizio (1976). By Frederick Novomestky.

gsarima Functions for Generalized SARIMA time series simulation. Write SARIMA models in (finite) AR representation and simulate generalized multiplicative seasonal autoregressive moving average (time) series with Normal/Gaussian, Poisson or negative binomial distribution. By Olivier Briet.

helloJavaWorld Hello Java World. A dummy package to demonstrate how to interface to a jar file that resides inside an R package. By Tobias Verbeke.

hsmm Computation of Hidden Semi Markov Models. By Jan Bulla, Ingo Bulla, Oleg Nenadic.

hydrogeo Groundwater data presentation and interpretation. Contains one function for drawing Piper (also called Piper-Henn) digrammes from water analysis for major ions. By Myles English.

hypergeo The hypergeometric function for complex numbers. By Robin K. S. Hankin.

ivivc A menu-driven package for in vitro-in vivo correlation (IVIVC) model building and model validation. By Hsin Ya Lee and Yung-Jin Lee.

jit Enable just-in-time (JIT) compilation. The functions in this package are useful only under Ra and have no effect under R. See `http://www.milbo.users.sonic.net/ra/index.html`. By Stephen Milborrow.

kerfdr Semi-parametric kernel-based approaches to local fdr estimations useful for the testing of multiple hypothesis (in large-scale genetic, genomic and post-genomic studies for instance). By M Guedj and G Nuel, with contributions from S. Robin and A. Celisse.

knorm Knorm correlations between genes (or probes) from microarray data obtained across multiple biologically interrelated experiments. The Knorm correlation adjusts for experiment dependencies (correlations) and reduces to the Pearson coefficient when experiment dependencies are absent. The Knorm estimation approach can be generally applicable to obtain between-row correlations from data matrices with two-way dependencies. By Siew-Leng Teng.

lago An efficient kernel algorithm for rare target detection and unbalanced classification. LAGO is a kernel method much like the SVM, except that it is constructed without the use of any iterative optimization procedure and hence very efficient (Technometrics 48, 193–205; The American Statistician 62, 97–109, Section 4.2). By Alexandra Laflamme-Sanders, Wanhua Su, and Mu Zhu.

latentnetHRT Latent position and cluster models for statistical networks. This package implements the original specification in Handcock, Raftery and Tantrum (2007) and corresponds to version 0.7 of the original **latentnet**. The

current package **latentnet** implements the new specification in Krivitsky and Handcock (2008), and represents a substantial rewrite of the original package. Part of the "statnet" suite of packages for network analysis. By Mark S. Handcock, Jeremy Tantrum, Susan Shortreed, and Peter Hoff.

limSolve Solving linear inverse models. Functions that (1) Find the minimum/maximum of a linear or quadratic function: min or max($f(x)$), where f(x) = $||Ax - b||^2$ or $f(x) = \sum a_i x_i$ subject to equality constraints $Ex = f$ and/or inequality constraints $Gx >= h$. (2) Sample an under-determined or over-determined system $Ex = f$ subject to $Gx >= h$, and if applicable $Ax = b$. (3) Solve a linear system $Ax = B$ for the unknown x. Includes banded and tridiagonal linear systems. The package calls Fortran functions from LINPACK. By Karline Soetaert, Karel Van den Meersche, and Dick van Oevelen.

lnMLE Maximum likelihood estimation of the Logistic-Normal model for clustered binary data. S original by Patrick Heagerty, R port by Bryan Comstock.

locpol Local polynomial regression. By Jorge Luis Ojeda Cabrera.

logregperm A permutation test for inference in logistic regression. The procedure is useful when parameter estimates in ordinary logistic regression fail to converge or are unreliable due to small sample size, or when the conditioning in exact conditional logistic regression restricts the sample space too severely. By Douglas M. Potter.

marginTree Margin trees for high-dimensional classification, useful for more than 2 classes. By R. Tibshirani.

maxLik Tools for Maximum Likelihood Estimation. By Ott Toomet and Arne Henningsen.

minet Mutual Information NEtwork Inference. Implements various algorithms for inferring mutual information networks from data. All the algorithms compute the mutual information matrix in order to infer a network. Several mutual information estimators are implemented. By P. E. Meyer, F. Lafitte, and G. Bontempi.

mixdist Contains functions for fitting finite mixture distribution models to grouped data and conditional data by the method of maximum likelihood using a combination of a Newton-type algorithm and the EM algorithm. By Peter Macdonald, with contributions from Juan Du.

moduleColor Basic module functions. Methods for color labeling, calculation of eigengenes, merging of closely related modules. By Peter Langfelder and Steve Horvath.

mombf This package computes moment and inverse moment Bayes factors for linear models, and approximate Bayes factors for GLM and situations having a statistic which is asymptotically normally distributed and sufficient. Routines to evaluate prior densities, distribution functions, quantiles and modes are included. By David Rossell.

moonsun A collection of basic astronomical routines for R based on "Practical astronomy with your calculator" by Peter Duffet-Smith. By Lukasz Komsta.

msProcess Tools for protein mass spectra processing including data preparation, denoising, noise estimation, baseline correction, intensity normalization, peak detection, peak alignment, peak quantification, and various functionalities for data ingestion/conversion, mass calibration, data quality assessment, and protein mass spectra simulation. Also provides auxiliary tools for data representation, data visualization, and pipeline processing history recording and retrieval. By Lixin Gong, William Constantine, and Alex Chen.

multipol Various utilities to manipulate multivariate polynomials. By Robin K. S. Hankin.

mvna Computes the Nelson-Aalen estimator of the cumulative transition hazard for multistate models. By Arthur Allignol.

ncf Functions for analyzing spatial (cross-) covariance: the nonparametric (cross-) covariance, the spline correlogram, the nonparametric phase coherence function, and related. By Ottar N. Bjornstad.

netmodels Provides a set of functions designed to help in the study of scale free and small world networks. These functions are high level abstractions of the functions provided by the **igraph** package. By Domingo Vargas.

networksis Simulate bipartite graphs with fixed marginals through sequential importance sampling, with the degrees of the nodes fixed and specified. Part of the "statnet" suite of packages for network analysis. By Ryan Admiraal and Mark S. Handcock.

neuralnet Training of neural networks using the Resilient Backpropagation with (Riedmiller, 1994) or without Weightbacktracking (Riedmiller, 1993) or the modified globally convergent version by Anastasiadis et. al. (2005). The package

allows flexible settings through custom choice of error and activation functions. Furthermore the calculation of generalized weights (Intrator & Intrator, 1993) is implemented. By Stefan Fritsch and Frauke Guenther, following earlier work by Marc Suling.

nlrwr Data sets and functions for non-linear regression, supporting software for the book "Nonlinear regression with R". By Christian Ritz.

nls2 Non-linear regression with brute force. By G. Grothendieck.

nlt A nondecimated lifting transform for signal denoising. By Marina Knight.

nlts functions for (non)linear time series analysis. A core topic is order estimation through cross-validation. By Ottar N. Bjornstad.

noia Implementation of the Natural and Orthogonal InterAction (NOIA) model. The NOIA model, as described extensively in Alvarez-Castro & Carlborg (2007, Genetics 176(2):1151-1167), is a framework facilitating the estimation of genetic effects and genotype-to-phenotype maps. This package provides the basic tools to perform linear and multilinear regressions from real populations (provided the phenotype and the genotype of every individuals), estimating the genetic effects from different reference points, the genotypic values, and the decomposition of genetic variances in a multi-locus, 2 alleles system. By Arnaud Le Rouzic.

normwn.test Normality and white noise testing, including omnibus univariate and multivariate normality tests. One variation allows for the possibility of weak dependence rather than independence in the variable(s). Also included is an univariate white noise test where the null hypothesis is for "white noise" rather than "strict white noise". The package deals with similar approaches to testing as the **nortest**, **moments**, and **mvnormtest** packages in R. By Peter Wickham.

npde Routines to compute normalized prediction distribution errors, a metric designed to evaluate non-linear mixed effect models such as those used in pharmacokinetics and pharmacodynamics. By Emmanuelle Comets, Karl Brendel and France Mentré.

nplplot Plotting non-parametric LOD scores from multiple input files. By Nandita Mukhopadhyay and Daniel E. Weeks.

obsSens Sensitivity analysis for observational studies. Observational studies are limited in that there could be an unmeasured variable related to both the response variable and the primary predictor. If this unmeasured variable were included in the analysis it would change the relationship (possibly changing the conclusions). Sensitivity analysis is a way to see how much of a relationship needs to exist with the unmeasured variable before the conclusions change. This package provides tools for doing a sensitivity analysis for regression (linear, logistic, and Cox) style models. By Greg Snow.

ofw Implements the stochastic meta algorithm called Optimal Feature Weighting for two multiclass classifiers, CART and SVM. By Kim-Anh Le Cao and Patrick Chabrier.

openNLP An interface to openNLP (`http://opennlp.sourceforge.net/`), a collection of natural language processing tools including a sentence detector, tokenizer, pos-tagger, shallow and full syntactic parser, and named-entity detector, using the Maxent Java package for training and using maximum entropy models. By Ingo Feinerer.

openNLPmodels English and Spanish models for **openNLP**. By Ingo Feinerer.

pga An ensemble method for variable selection by carrying out Darwinian evolution in parallel universes. PGA is an ensemble algorithm similar in spirit to AdaBoost and random forest. It can "boost up" the performance of "bad" selection criteria such as AIC and GCV. (Technometrics 48, 491–502; The American Statistician 62, 97–109, Section 4.3). By Dandi Qiao and Mu Zhu.

phangorn Phylogenetic analysis in R (estimation of phylogenetic trees and networks using maximum likelihood, maximum parsimony, distance methods & Hadamard conjugation). By Klaus Schliep.

plotSEMM Graphing nonlinear latent variable interactions in SEMM. Contains functions which generate the diagnostic plots proposed by Bauer (2005) to investigate nonlinear latent variable interactions in SEMM using LISREL output. By Bethany E. Kok, Jolynn Pek, Sonya Sterba and Dan Bauer.

poilog Functions for obtaining the density, random deviates and maximum likelihood estimates of the Poisson log-normal distribution and the bivariate Poisson log-normal distribution. By Vidar Grøtan and Steinar Engen.

prob Provides a framework for performing elementary probability calculations on finite sample spaces, which may be represented by data frames or lists. Functionality includes setting up sample spaces, counting tools, defining

probability spaces, performing set algebra, calculating probability and conditional probability, tools for simulation and checking the law of large numbers, adding random variables, and finding marginal distributions. By G. Jay Kerns.

profileModel Tools that can be used to calculate, evaluate, plot and use for inference the profiles of *arbitrary* inference functions for *arbitrary* `glm`-like fitted models with linear predictors. By Ioannis Kosmidis.

profr An alternative data structure and visual rendering for the profiling information generated by `Rprof`. By Hadley Wickham.

qAnalyst Control charts for variables and attributes according to the book "Introduction to Statistical Quality Control" by Douglas C. Montgomery. Moreover, capability analysis for normal and non-normal distributions is implemented. By Andrea Spanó and Giorgio Spedicato.

qpcR Model fitting, optimal model selection and calculation of various features that are essential in the analysis of quantitative real-time polymerase chain reaction (qPCR). By Andrej-Nikolai Spiess and Christian Ritz.

r2lUniv R to LaTeX Univariate. Performs some basic analysis and generate the corresponding LaTeX code. The basic analysis depends of the variable type (nominal, ordinal, discrete, continuous). By Christophe Genolini.

randomLCA Fits random effects latent class models, as well as standard latent class models. By Ken Beath.

richards Fit Richards curves. By Jens Henrik Badsberg.

risksetROC Compute time-dependent incident/dynamic accuracy measures (ROC curve, AUC, integrated AUC) from censored survival data under proportional or non-proportional hazard assumption of Heagerty & Zheng (2005, Biometrics 61:1, 92–105). By Patrick J. Heagerty, packaging by Paramita Saha.

robfilter A set of functions to filter time series based on concepts from robust statistics. By Roland Fried and Karen Schettlinger.

s20x Stats 20x functions. By Andrew Balemi, James Curran, Brant Deppa, Mike Forster, Michael Maia, and Chris Wild.

sampleSelection Estimation of sample selection models. By Arne Henningsen and Ott Toomet.

scrime Tools for the analysis of high-dimensional data developed/implemented at the group "Statistical Complexity Reduction In Molecular Epidemiology" (SCRIME). Main focus is on SNP data, but most of the functions can also be applied to other types of categorical data. By Holger Schwender and Arno Fritsch.

segclust Segmentation and segmentation/clustering. Corresponds to the implementation of the statistical model described in Picard et. al., "A segmentation/clustering model for the analysis of array CGH data" (2007, Biometrics, 63(3)). Segmentation functions are also available (from Picard et al., "A statistical approach for array CGH data analysis" (2005, BMC Bioinformatics 11;6:27)). By Franck Picard.

shape Plotting functions for creating graphical shapes such as ellipses, circles, cylinders, arrows, and more. Support for the book "A guide to ecological modelling" by Karline Soetaert and Peter Herman (in preparation). By Karline Soetaert.

siar Stable Isotope Analysis in R. This package takes data on organism isotopes and fits a Bayesian model to their dietary habits based upon a Gaussian likelihood with a mixture Dirichlet-distributed prior on the mean. By Andrew Parnell.

similarityRichards Computing and plotting of values for similarity of backfitted independent values of Richards curves. By Jens Henrik Badsberg.

space Partial correlation estimation with joint sparse regression model. By Jie Peng, Pei Wang, Nengfeng Zhou, and Ji Zhu.

stab A menu-driven package for data analysis of drug stability based on ICH guideline (such as estimation of shelf-life from a 3-batch profile.). By Hsin-ya Lee and Yung-jin Lee.

statnet An integrated set of tools for the representation, visualization, analysis and simulation of network data. By Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, Martina Morris.

subplex The subplex algorithm for unconstrained optimization, developed by Tom Rowan. By Aaron A. King, Rick Reeves.

survivalROC Compute time-dependent ROC curve from censored survival data using Kaplan-Meier (KM) or Nearest Neighbor Estimation (NNE) method of Heagerty, Lumley & Pepe (2000, Biometrics 56:2, 337–344). By Patrick J. Heagerty, packaging by Paramita Saha.

torus Torus algorithm for quasi random number generation (for Van Der Corput low-discrepancy sequences, use **fOptions** from Rmetrics). Also implements a general linear congruential pseudo random generator (such as Park Miller) to make comparison with Mersenne Twister (default in R) and the Torus algorithm. By Christophe Dutang and Thibault Marchal.

tpr Regression models for temporal process responses with time-varying coefficient. By Jun Yan.

xts Extensible Time Series. Provide for uniform handling of R's different time-based data classes by extending **zoo**, maximizing native format information preservation and allowing for user level customization and extension, while simplifying cross-class interoperability. By Jeffrey A. Ryan and Josh M. Ulrich.

yaml Methods to convert R to YAML and back, implementing the Syck YAML parser (`http://www.whytheluckystiff.net/syck`) for R. By Jeremy Stephens.

Other changes

- New task views `Optimization` (packages which offer facilities for solving optimization problems, by Stefan Theussl) and `ExperimentalDesign` (packages for experimental design and analysis of data from experiments, by Ulrike Groemping).
- Packages **JLLprod**, **butler**, **elasticnet**, **epsi**, **gtkDevice**, **km.ci**, **ncvar**, **riv**, **rpart.permutation**, **rsbml**, **taskPR**, **treeglia**, **vardiag** and **zicounts** were moved to the Archive.
- Package **CPGchron** was moved to the Archive (replaced by **Bchron**).
- Package **IPSUR** was moved to the Archive (replaced by **RcmdrPlugin.IPSUR**).
- Package **gRcox** was renamed to **gRc**.
- Package **pwt** was re-added to CRAN.

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
`Kurt.Hornik@R-project.org`

News from the Bioconductor Project

by the Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center

We are pleased to announce Bioconductor 2.2, released on May 1, 2008. Bioconductor 2.2 is compatible with R 2.7.0, and consists of 260 package. The release includes 35 new packages, and many improvements to existing packages.

New packages

New packages address a diversity of topics in high-throughput genomic analysis. Some highlights include:

Advanced statistical methods for analysis ranging from probe-level modeling (e.g., plw) through gene set and other functional profiling (e.g., GSEAlm, goProfiles).

New problem domains addressed by packages such as snpMatrix, offering classes and methods to compactly summarize large single nucleotide polymorphism data sets.

Integration with third-party software including the GEOmetadb package for accessing GEO metadata and AffyCompatible and additions to affxparser for accessing microarray vendor resources.

Graphical tools in packages such as GenomeGraphs and rtracklayer effectively visualize complex data in an appropriate genomic context.

New technical approaches in packages such as affyPara and xps explore the significant computational burden of large-scale analysis.

The release also includes packages to support two forthcoming books: by Gentleman (2008), about using R for bioinformatics; and by Hahne et al. (2008), presenting Bioconductor case studies.

Annotations

The 'annotation' packages in Bioconductor have experienced significant change. An annotation package contains very useful biological information about microarray probes and the genes they are meant to interrogate. Previously, these packages used an R environment to provide a simple key-value association between the probes and their annotations. This release of Bioconductor sees widened use of SQLite-based annotation packages, and SQLite-based annotations can now be used instead of most environment-based packages.

SQLite-based packages offer several attractive features, including more efficient use of memory, representation of more complicated data structures, and flexible queries across annotations (SQL tables). Most users access these new annotations using familiar functions such as `mget`. One useful new function is the `revmap` function, which has the effect (but not the overhead!) of reversing the direction of the map (e.g., mapping from gene symbol to probe identifier, instead of the other way around). Advanced users can write SQL queries directly.

The scope of annotation packages continues to expand, with a more extensive 'organism'-centric (e.g., org.Hs.eg.db, representing *Homo sapiens*) annotations. New 'homology' packages summarize the InParanoid data base, allowing between-species identification of homologous genes.

Other developments and directions

Bioconductor package authors continue to have access to a very effective package repository and build system. All packages are maintained under subversion version control, with the latest version of the package built each day on a diversity of computer architectures. Developers can access detailed information on the success of their package builds on both release and development platforms (e.g., `http://bioconductor.org/checkResults/`). Users access successfully built packages using the `biocLite` function, which identifies the appropriate package for their version of R.

New Bioconductor packages contributed from our active user / developer base now receive both technical and scientific reviews. This helps package authors produce quality packages, and benefits users by providing a more robust software experience.

The 2.3 release of Bioconductor is scheduled for October 2008. We expect this to be a vibrant release cycle. High-throughput genomic research is a dynamic and exciting field. It is hard to predict what surprising packages are in store for future Bioconductor releases. We anticipate continued integration with diverse data sources, use of R's advanced graphics abilities, and implementation of cutting edge research algorithms for the benefit of all Bioconductor users. Short-read DNA resequencing technologies are one area where growth seems almost certain.

Bibliography

R. Gentleman. *Bioinformatics with R*. Chapman & Hall/CRC, Boca Raton, FL, 2008. ISBN 1-420-06367-7.

F. Hahne, W. Huber, R. Gentleman, and S. Falcon. *Bioconductor Case Studies*. Springer, 2008.

Forthcoming Events: useR! 2008

The international R user conference 'useR! 2008' will take place at the Technische Universität Dortmund, Dortmund, Germany, August 12-14, 2008.

This world-wide meeting of the R user community will focus on

- R as the 'lingua franca' of data analysis and statistical computing;
- providing a platform for R users to discuss and exchange ideas about how R can be used to do statistical computations, data analysis, visualization and exciting applications in various fields;
- giving an overview of the new features of the rapidly evolving R project.

The program comprises invited lectures, user-contributed sessions and pre-conference tutorials.

Invited Lectures

R has become the standard computing engine in more and more disciplines, both in academia and the business world. How R is used in different areas will be presented in invited lectures addressing hot topics. Speakers will include

- *Peter Bühlmann*: Computationally Tractable Methods for High-Dimensional Data
- *John Fox* and *Kurt Hornik*: The Past, Present, and Future of the R Project, a double-feature presentation including
  Social Organization of the R Project (John Fox), Development in the R Project (Kurt Hornik)
- *Andrew Gelman*: Bayesian Generalized Linear Models and an Appropriate Default Prior
- *Gary King*: The Dataverse Network
- *Duncan Murdoch*: Package Development in Windows
- *Jean Thioulouse*: Multivariate Data Analysis in Microbial Ecology – New Skin for the Old Ceremony
- *Graham J. Williams*: Deploying Data Mining in Government – Experiences With R/Rattle

User-contributed Sessions

The sessions will be a platform to bring together R users, contributors, package maintainers and developers in the S spirit that 'users are developers'. People from different fields will show us how they solve problems with R in fascinating applications. The scientific program is organized by members of the program committee, including *Micah Altman, Roger Bivand, Peter Dalgaard, Jan de Leeuw, Ramón Díaz-Uriarte, Spencer Graves, Leonhard Held, Torsten Hothorn, François Husson, Christian Kleiber, Friedrich Leisch, Andy Liaw, Martin Mächler, Kate Mullen, Ei-ji Nakama, Thomas Petzoldt, Martin Theus*, and *Heather Turner*, and will cover topics such as

- Applied Statistics & Biostatistics
- Bayesian Statistics
- Bioinformatics
- Chemometrics and Computational Physics
- Data Mining
- Econometrics & Finance
- Environmetrics & Ecological Modeling
- High Performance Computing
- Machine Learning
- Marketing & Business Analytics
- Psychometrics
- Robust Statistics
- Sensometrics
- Spatial Statistics
- Statistics in the Social and Political Sciences
- Teaching
- Visualization & Graphics
- and many more

Pre-conference Tutorials

Before the start of the official program, half-day tutorials will be offered on Monday, August 11.
In the morning:

- *Douglas Bates*: Mixed Effects Models
- *Julie Josse, François Husson, Sébastien Lê*: Exploratory Data Analysis
- *Martin Mächler, Elvezio Ronchetti*: Introduction to Robust Statistics with R
- *Jim Porzak*: Using R for Customer Segmentation
- *Stefan Rüping, Michael Mock, and Dennis Wegener*: Distributed Data Analysis Using R
- *Jing Hua Zhao*: Analysis of Complex Traits Using R: Case studies

In the afternoon:

- *Karim Chine*: Distributed R and Bioconductor for the Web
- *Dirk Eddelbuettel*: An Introduction to High-Performance R
- *Andrea S. Foulkes*: Analysis of Complex Traits Using R: Statistical Applications
- *Virgilio Gómez-Rubio*: Small Area Estimation with R
- *Frank E. Harrell, Jr.*: Regression Modelling Strategies
- *Sébastien Lê, Julie Josse, François Husson*: Multiway Data Analysis
- *Bernhard Pfaff*: Analysis of Integrated and Cointegrated Time Series

More Information

A web page offering more information on 'useR! 2008' as well as the registration form is available at:
`http://www.R-project.org/useR-2008/`.

We hope to meet you in Dortmund!

The organizing committee:
Uwe Ligges, Achim Zeileis, Claus Weihs, Gerd Kopp, Friedrich Leisch, and Torsten Hothorn
`useR-2008@R-project.org`

R Foundation News

by Kurt Hornik

Donations and new members

Donations

Austrian Association for Statistical Computing
Fabian Barth, Germany
Dianne Cook, USA
Yves DeVille, France
Zubin Dowlaty, USA
David Freedman, USA
Minato Nakazawa, Japan

New benefactors

Paul von Eikeren, USA
InterContinental Hotels Group, USA

New supporting institutions

European Bioinformatics Inst., UK

New supporting members

Simon Blomberg, Australia
Yves DeVille, France
Adrian A. Dragulescu, USA
Owe Jessen, Germany
Luca La Rocca, Italy
Sam Lin, New Zealand
Chris Moriatity, USA
Nathan Pellegrin, USA
Peter Ruckdeschel, Germany
Jitao David Zhang, Germany

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
`Kurt.Hornik@R-project.org`

R News Referees 2007

by John Fox

R News articles are peer-reviewed. The editorial board members would like to take the opportunity to thank all referees who read and commented on submitted manuscripts during the previous year. Much of the quality of R News publications is due to their invaluable and timely service. Thank you!

- Murray Aitkin
- Doug Bates
- Adrian Bowman
- Patrick Burns
- Peter Dalgaard
- Philip Dixon
- Dirk Eddelbuettel
- Brian Everitt
- Thomas Gerds
- B.J. Harshfield
- Sigbert Klinke
- Thomas Kneib
- Anthony Lancaster
- Duncan Temple Lang
- Thomas Lumley
- Martin Maechler
- Brian McArdle
- Georges Monette
- Paul Murrell
- Martyn Plummer
- Christina Rabe
- Alec Stephenson
- Carolin Strobl
- Simon Urbanek
- Keith Worsley

John Fox
McMaster University, Canada
John.Fox@R-project.org