The R Journal

Volume 1/2, December 2009

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

Contents

The R Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are copyrighted by the respective authors.

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

Editor-in-Chief:
Vince Carey
Channing Laboratory
Brigham and Women's Hospital
75 Francis St.
Boston, MA 02115 USA

Editorial Board:
John Fox, Heather Turner, and Peter Dalgaard.

Editor Programmer's Niche:
Bill Venables

Editor Help Desk:
Uwe Ligges

R Journal Homepage:
http://journal.r-project.org/

Email of editors and editorial board:
firstname.lastname@R-project.org

Editorial

by Vince Carey

This issue of the R Journal comes on the heels of R 2.10.1. R 2.10 sports a variety of changes to

- the documentation system
- factor handling
- debugging and code analysis support
- encodings management
- [[semantics
- regular expression processing
- data compression facilities
- package installation and checking

among other features. Most users will want to familiarize themselves with the details of items described in R_HOME/NEWS and in this issue's "Changes to R" article. Thanks are due to the core members and other contributors who have introduced these enhancements, many of which will increase the ease and scope of use of R in the growing set of domains for which effectiveness requires excellent data analysis.

The R Journal also has some new or impending features of interest. A number of readers have inquired about subscriptions and RSS feeds. We now have a feed, thanks to Heather Turner: http://journal.r-project.org/rss.xml. It is also a pleasure to announce the addition of Jay Kerns as Book Review Editor; the Book Review section will be inaugurated in the next issue. Thanks to efforts of Achim Zeileis, we have added subsections to the "Changes on CRAN" regular feature that describe new CRAN task views and new allocations of packages to CRAN task views. In the PDF image of the Journal, these are all hyperlinked to the view or package resources on CRAN, so that readers can quickly investigate or acquire packages in views of interest. Finally, in this issue we have a nice piece by R core members Duncan Murdoch and Simon Urbanek describing changes to the R help markup language and its processing. This article is the first for a recurring journal section "From the Core" where we plan to highlight new ideas and methods in the words of core members themselves.

It has been a pleasure to assemble this number. We have a special item on the sociology of the R project from our past editor-in-chief, John Fox. Research articles cover topics in random forest interpretation, meta-analysis, complex surveys in health policy research, data mining via GUI, enhanced support for resource discovery, and issues in teaching about convergence of sequences of random variables and large sample inference.

My tenure as Editor-in-Chief of the R Journal comes to a close with this issue. Peter Dalgaard now takes the reins. I am deeply indebted to Peter, John Fox, Heather Turner, Uwe Ligges, and Bill Venables for their editorial assistance, and to Martin Maechler for systems support. John Fox is owed a special thanks for staying in the editorial group for an extra year; we welcome Martyn Plummer of IARC who is joining as Associate Editor.

To close, I'd like to suggest to readers that they spend at least a little while in the "Changes on CRAN" section. There is much to be learned there from the perspective of software interoperability alone, with new packages defining interfaces to MS Word, Apache ant, NVIDIA CUDA, and sendmail, for example. Folks interested in working with AVIRIS hyperspectral images, NIfTI-formatted brain images, or the TikZ system for algebraically specified vector graphics will find connections to R in this section. Owners of multicore hardware will want to get acquainted with new contributions from Revolution Computing, Inc. Lastly, browsing the new contributions inspired me to learn that "quaternary science" denotes the study of the past 2.6 million years on Earth. Go CRAN!

Vince Carey
Channing Laboratory, Brigham and Women's Hospital
Vince.Carey@R-project.org

Aspects of the Social Organization and Trajectory of the R Project

by John Fox

Abstract: Based partly on interviews with members of the R Core team, this paper considers the development of the R Project in the context of open-source software development and, more generally, voluntary activities. The paper describes aspects of the social organization of the R Project, including the organization of the R Core team; describes the trajectory of the R Project; seeks to identify factors crucial to the success of R; and speculates about the prospects for R.

Introduction

This paper describes aspects of the R Core team; briefly traces the trajectory of the R Project; discusses the development and organization of the R Project; considers the reasons for the success of R; and speculates about its prospects for continued success. The paper is based on semi-structured interviews conducted during 2006 and 2007 with most members of the R Core team, whom I will occasionally quote in the paper; on publicly available archival sources; and on participant observation in the R Project, as a user, package developer, author, and — more recently — a member of the R Foundation.

The paper is not a complete consideration of the social organization of the R Project in that it does not systematically address interactions among members of the R Core team, nor between the Core team and package developers and users, nor among developers and users, all of which would be the proper subject of a more complete account. Nevertheless, I do try to identify key aspects of the social design of the R Project, particularly with respect to their contributions to the success of R and to its future.

What is problematic about open-source software development?

Why do people contribute to open-source projects such as R? Is this behaviour purely altruistic, or are there rewards — tangible and otherwise — to open-source development? Raymond (2001c), for example, suggests that the open-source development community constitutes a "gift culture" in which the currency is reputation; he also argues (Raymond, 2001d) for the economic rationality of businesses that support open-source development. Similar arguments about the potential rationality of participation in open-source projects are advanced by Weber (2004).

Although I will address the question of motivation briefly (and although it is raised repeatedly by economists), it is not an issue unique to open-source software development: After all, people participate in a wide variety of voluntary organizations. There is a large and venerable literature in sociology on voluntary associations (for reviews, see Smith, 1975, and Knoke, 1986), much of it focusing on participation, and more recent work in the area addressing the "social capital" accruing to communities as a consequence of participation in voluntary organizations (following Putnam, 1995).

Winchester (2003, p. 215) writes of the unpaid volunteers who contributed meticulous work to the monumental *Oxford English Dictionary*:

> [W]e do not really know why so many people gave so much of their time for so little apparent reward. And this is the abiding and most marvelous mystery of the enormously democratic process that was the Dictionary — that hundreds upon hundreds of people, for motives known and unknown, for reasons both stated and unsaid, helped to chronicle the immense complexities of the language that was their own, and that they dedicated in many cases ... years upon years of labour to a project of which they all, buoyed by some set of unfathomable and optimistic notions, insisted on becoming a part.

With a few changes in specifics, much the same can be said of participation in the R Project — both by members of the Core team and by others.

Participation in open-source software projects is in this sense no different from participation in other voluntary organizations, such as coaching a children's ice-hockey team or contributing to the *OED*. When asked about their motivation for working on the R project, members of the R Core team responded with conventional reasons for participating in a voluntary association:

- To satisfy a sense of obligation (with a hint of rational self-interest).

> [M]y feeling is that I gain great benefit from open-source software. This is tremendously valuable to me, being able to use all of these other tools, and I feel both a moral and practical obligation to contribute back into this sea of tools that are, I

> think, very important for the development of our profession.

And, in another case:

> [P]robably more than half of what I've done for the R Project is not because I needed it, but because I thought that the R Project needed it. There are relatively few things in R that I have done that I needed

- To do intrinsically rewarding work.

> [It's] very satisfying ...to work on a day-to-day basis with people with whom one has common interests and can get a lot of pleasure from working with.

- To contribute to the broader public good.

> One of the nicest sort of things [is that] other people in the Philippines or Bolivia or Mexico ...can have a world class statistical software system [when] they could never afford any of the commerical systems.

In another case:

> [Y]ou want to actually contribute something for the greater good. I mean that's why we do this stuff — in the hope that it is actually of importance to some people, [that] it's going to change things.

What seems to me most problematic about open-source software development, however, is *how* the work gets done. Traditionally, software was developed by rigidly hierarchical organizations that reflected the perceived necessity for an overall design of a software project and a division of labour in its execution. Despite the ideology of open-source development as a "bazaar" (Raymond, 2001b), an open-source software project of any complexity must (like the *OED*) come to grips with the problems of social organization, including direction and division of labour, if it is to succeed. After all, different parts of the software have to combine in a functional whole.

The R Core team

The R Core team is responsible for the development of the basic R software and for the infrastructure that supports its continued development and distribution. As most readers of this journal are likely aware, the R Project began around 1990 as the informal endeavour of Robert Gentleman and Ross Ihaka, who were then both at Auckland University in New Zealand. The initial development of R is described in Ihaka and Gentleman (1996). Several other individuals became part of the project, in the sense of having write-access to the R source code, over the next five or six years, but the organization of the project remained informal. Then, in 1997, this structure was formalized in the creation of an R Core group with nine members, a number that has subsequently grown to 19 (see Figure 1).

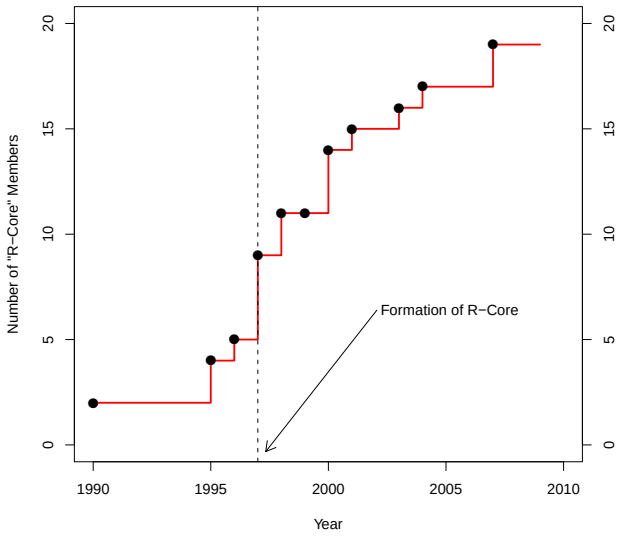

Figure 1: Growth of the R Core team. Points represent changes in membership. *Source of data:* Interviews and personal communications with members of R Core.

Open-source software projects vary a great deal in their organization: Some are undertakings of large corporations, such as Sun (Open Office), IBM (the Eclipse interactive software development environment), or Google (the Android smart-phone platform). Many open-source projects, such as the Linux operating system, have strongly hierarchical structures, and some, such as Perl, revolve around a central individual. (See, e.g., Fogel, 2006, for a prescriptive treatment of the organization of open-source software projects, along with the ideological writings of Raymond, 2001a, and Stallman, 2002.) Although I haven't formally surveyed the many existing open-source projects, the R Project is apparently unusual in its flat formal organization of independent volunteers. This structure has attracted a remarkably talented and competent group of individuals, but, as I will argue later in this paper, it poses challenges for moving the R Project forward.

Although there are few formally differentiated roles among members of the R Core team, there is a fuzzy division of labour. Certain members of R Core, for example, are responsible for maintaining the versions of R for different operating systems,

and other members for maintaining the CRAN package archive. The development of this division of labour was essentially accidental, particularly in the early stages of the R Project. More recently, several members of the R Core team were recruited at least partly because their skills and expertise complemented those of then-current members of the Core group.

A loose, more-or-less naturally developed division of labour serves the day-to-day needs of the R Project — that is, routine tasks such as preparation and dissemination of a new release, making marginal improvements to R, or fixing bugs that are discovered in the software — but it does not account for new directions in the development of R, particularly when decisions are controversial.

According to my interviews with R Core members, the Core group operates according to what I would describe as a modified-consensus model. An issue is discussed, for example, at occasional face-to-face meetings of R Core members and more frequently by email. An attempt is made to reach a consensus, but consensus is not always possible. In such cases, frequently nothing is done to move the issue forward, and development simply does not take place. In other instances, development is pushed forward preemptively by one or a small number of R Core members who are willing simply to implement a change. One example that came up repeatedly in my interviews was the implementation of internationalization in R. As one member of R Core put it to me, "[We have] a system that [is] democratic but the person who [is] going to do the work [gets] more votes than anybody else." In such cases, the silence of other R Core members connotes consent — or at least acquiescence. Another R Core member said, "[C]onsensus could be as simple as nobody objecting." Except with respect to membership in the Core group, formal votes do not occur.

A number of members of R Core referred to a more formal process by which at least some larger innovations are first described in written proposals:

> [F]or more complex things, people understand they need to write a request for comments, which they do, and that generates its own thread; and on the basis of that, the proposal is either modified or, very rarely, things have to be abandoned.

Table 1 attempts to summarize several key aspects of the organization of the R Project during three "stages" of its development: (1) an initial stage, during which R was the exclusive project of Gentleman and Ihaka, with some student participation; (2) a transitional stage during which several other developers were recruited to the project; and (3) a continuing mature stage during which the R Project has been guided by a formally constituted Core team.

Figure 2 points out another characteristic of the R Core group — that activity in the R svn (version-control) archive is highly unequal, and that the level of inequality has increased since 1997. I hasten to add that this activity, assessed as number of "commits" to the archive by each member of R Core, is measured imperfectly: Individual "commits" to the archive can represent vastly different amounts of code, and the contribution of an R Core member to the project may not be reflected in changes to the svn archive. Nevertheless, the Gini coefficient for commits by R Core members has risen from 0.67 to 0.85 in a little more than a decade, and the proportion of commits by the single most active individual (not the same person, by the way, through this period) reached a high of 0.73 in 2007.

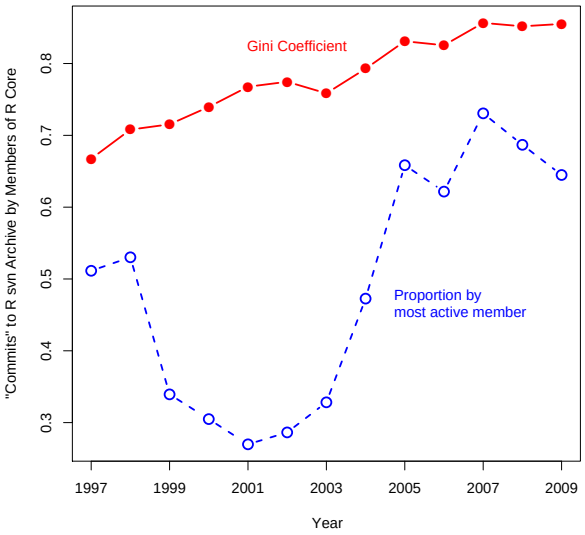

Figure 2: There is a high level of inequality among R Core members in the number of "commits" to the R svn archive. *Source of data:* `http://developer.r-project.org/`.

The trajectory of the R Project

Writing about R in a book published in 2002, I pointed approvingly to the "more than 100 contributed packages available on the R Web site ... many of them prepared by experts in various areas of applied statistics." Comparing R to S-PLUS, I said that, "I believe that the current development of R is more dynamic." There are now nearly 2000 contributed packages on CRAN. The left panel of Figure 3 shows that the growth in CRAN packages has been approximately exponential — the linear correlation between the log of the number of packages and time exceeds 0.99; a plot of residuals, however, in the right panel of Figure 3, shows that the rate of growth has been slowing recently.

Although it is more difficult to assess how widely

	Stage		
	Initial	*Transitional*	*R Core*
Approximate Dates	1990-94	1994-97	1997-
Recruitment	some student participation	demonstrated interest	semi-purposive, by invitation
Division of labour	none	developing	semi-formal
Hierarchy	none	original developers, contributors	differential participation
Principal Mode of Cooperation	direct collaboration	anarchic voluntarism	partly distinct roles + voluntarism
Planning	none	implicit	partial
Decision-Making	joint	individual	modified consensus
Resolution of Disagreements	discussion	largely unnecessary	discussion, preemption, avoidance
principal goal	personal development	reproduce and improve S	various, partly conflicting

Table 1: Stages in the development of the R Project.

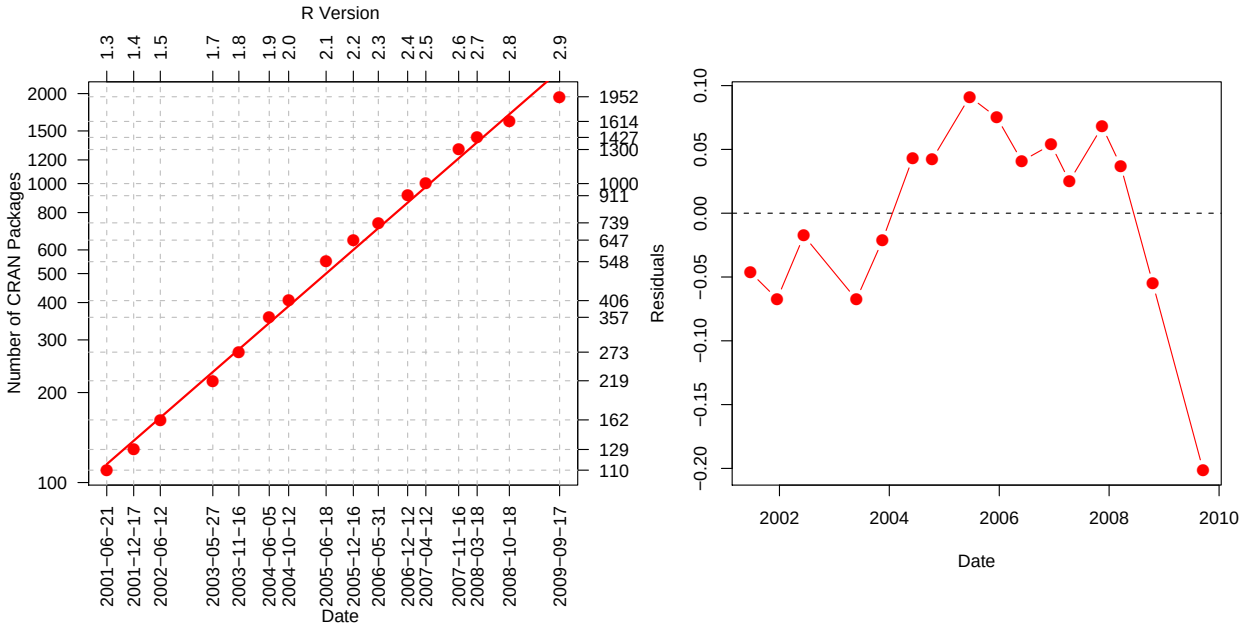

Figure 3: The number of packages on CRAN (left panel) has grown roughly exponentially, with residuals from the exponential trend (right panel) showing a recent decline in the rate of growth. The number of packages for R version 1.6 is not shown because the count was taken only two days after that for version 1.5, and therefore indicated just one additional package. (An earlier version of the graph in the left panel appeared in Fox, 2008.) *Sources of data:* `https://svn.r-project.org/R/branches/` and (for version 2.9) `http://cran.r-project.org/web/checks/check_summary.html`.

R is used, the anecdotal evidence makes the hypothesis of exponential growth coupled with currently wide use plausible. A recent article in both the print and on-line editions of *The New York Times* (Vance, 2009) supports the increasing popularity of R — regardless of the merits of the article itself. Likewise, developers of commercial statistical software, such as SPSS and SAS, who find themselves in competition with R, have moved to make R available from within their products. Figure 4 shows the monthly rate of messages on the r-help, r-devel, and other ("SIG") email lists from 1997 to 2008. After a period of less-than-exponential growth, the growth in the overall number of messages appears once more to be accelerating.

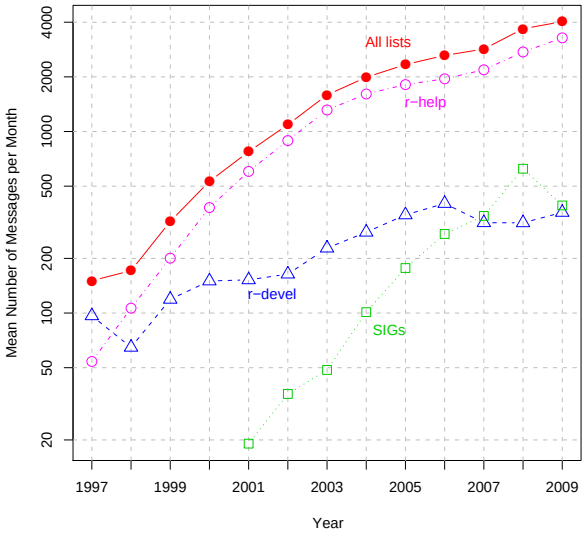

Figure 4: There has been dramatic growth in traffic on the R email lists. *Source of data:* `http://www.r-project.org/mail.html`.

Why did the R Project succeed?

That the R Project has "succeeded" is undeniable, and the members of the R Core team whom I interviewed unanimously endorsed that judgment. This success rests on a number of factors (some of which I have already mentioned):

Open-source development

The initial developers — Robert Gentleman and Ross Ihaka — opened up the project, eventually forming the R Core group and releasing R under the GNU General Public License (GPL). Other software, such as Octave (Eaton et al., 2008) and Lisp-Stat (Tierney, 1990), which followed a different development model focusing on a single key individual, have not gained as wide acceptance.[1]

Partly by accident, but increasingly by design, the R Project attracted a Core group of immensely talented individuals, including leading figures in statistical computing.

The S language

After an initial period in which it served mainly to satisfy the curiosity of its creators, the R Project acquired a concrete, if not exclusive, target: roughly to reproduce the functionality of S in a language with essentially similar syntax.[2]

As a corollary to the last point, much of the necessary software beyond the basic R system was already available in S "libraries" (e.g., **MASS**, Venables and Ripley, 2002; **survival**, Therneau and Lumley, 2009; **nlme**, Pinheiro et al., 2009) which could be ported to R. As an additional corollary, the S language had substantially penetrated the community of statisticians, and so R had a favourably disposed, strategically important group of potential users.

In comparison to other free statistical software (e.g., Lisp-Stat) R is relatively easy to use, particu-

[1] I base this claim partly on my interviews with members of R Core who had experience with other software. Note that the point is independent of the intentions of the original developers of R and other software, but is rather meant to be descriptive of the effects of the manner in which they incorporated contributions from other people.

[2] Although this assertion was disputed by a referee, I think that an examination of the contents of the standard R distribution supports it. Moreover, the point came up several times in my interviews; for example,

> [T]he rough idea that the R people have had over the years is that the ... so-called Blue Book and ... the so-called White Book are kind of a rough definition of the S language. And so, not necessarily always, but fairly often, if a question comes up about what something should do or how it should be interpreted there's a tendency to go back to those books and look at them.

Another member of R Core said,

> R wasn't created from scratch — well, it was, but there was a model, some members of the core team call it the prototype. So when there were disputes about the way things should work, then the development process was inevitably drawn towards similar behaviour to S, except in cases where the S behaviour was obviously wrong. ... [C]ompatibility with S ... became an important design requirement. When R first started it was ... in many ways quite different.

Just to be clear: I don't mean to claim that R simply or exclusively reproduces S, nor that replicating the functionality of S was an initial goal of the original R duo of developers. As well, I'm focusing here on the syntax of R — which is the feature of the language most visible initially to users — as opposed to its semantics, which derive from the Scheme dialect of Lisp (see Ihaka and Gentleman, 1996).

[3] Of course, those previously familiar with Lisp will have little difficulty picking up Lisp-Stat, but a fair comparison assumes a user without this sort of background. My own experience provides some anecdotal support for the assertion: I came to R directly from APL and Lisp-Stat. I also had some prior, superficial exposure to S, had programmed in several different languages starting with Fortran, and used a variety of statistical packages, such as BMDP, Minitab, SPSS, and SAS. It took me at least a couple of months of intensive work

larly for those already familiar with S.[3] I'm aware that there are frequent remarks about the "steep learning curve" associated with R, but I think that these complaints have point-and-click statistical software and not competing statistical programming environments as their primary reference.

The R package system

The package system, introduced early in the life of R, permits individuals to participate in the development of R without the direct intervention of the R Core group. In a sense, the package system — like version control — is a technological solution to a social problem: how to invite, motivate, and coordinate the activity of hundreds of volunteers without overwhelming the resources of the Core team. As one R Core member put it,

> [T]he package system allows us to take advantage of lots of people without having to find some way of getting consensus from them. ...[The package system] wasn't designed with that social purpose in mind but I think that it's been critical in R being successful.

The package system provides a variety of integrative functions, including quality control; enforcement of standards; provision of a common documentation format, which itself has certain novel and noteworthy features (e.g., the incorporation of executable examples); and convenient distribution.

In addition to help files, the package system supports the provision of automatically-compiled longer documents, termed "vignettes." More generally, R has cross-fertilized the development of literate programming tools for statistics, most notably Sweave (described in Leisch, 2002, 2003).

Because R is programmable, it permits users to develop software for their own use. The package system encourages them then to share this software with others and, to a limited extent, facilitates recognition for software development. This process allows R to grow in a natural, organic manner. Although R is not unique in providing a programming environment for statistical applications, nor in providing a mechanism for sharing code, the R package system is particularly well worked out. This general point, moreover, extends to the the core developers as well, whose work on R is partly motivated by their research interests, both statistical and substantive. The availability of a wide variety of contributed packages, in turn, enhances the attractiveness of R to a diverse group of users, including those who do not write their own programs

The package system also serves at least partly to circumvent disputes that might otherwise fracture the R Project. Although this route has its limitations, one can innovate in packages without making changes directly to the basic R system. As has been pointed out to me, however, the same might be said of other open-source software projects that support packages, and some of these projects — for example, Emacs and Linux — have forked. But the general point seems sound: A successful package system raises the threshold for forking an open-source project.

Other considerations

Several other factors have contributed to R's success:

- As is common in open-source software projects, the R Core group has successfully leveraged information technology, both to coordinate its own activities (e.g., by version control) and to involve others in the project (e.g., via e-mail lists, package automation, and distribution via the Internet). Many of the members of R Core, however, stressed the importance of periodic face-to-face meetings among the core developers. One said, for example, that face-to-face meetings were "crucial," and continued,

  > [Y]ou can sort out details by e-mail, but you cannot thrash out general policy, and so having a meeting every year or two is really important to get it together and get general ideas sorted out.

- R runs on all widely used computational platforms (Windows, Mac OS X, and Linux/Unix systems).

- Where R differs from S, it has clearly improved on the latter. Some examples are lexical scoping, the package system, and the introduction of namespaces. Development of new software in R, therefore, has become more attractive than in S-PLUS, even when cost is not an issue.

- And, of course, R is free, in both of Stallman's (2002) senses — R is available free of cost, and users of R are free to examine, modify, and redistribute the software.

What are the prospects for R?

Many of the factors leading to the initial success of R are of continued relevance. In addition, R has accumulated a great deal of momentum: It has attracted

to become reasonably comfortable with Lisp, but just a few days to reach a similar level of competence in R (and I don't think that the transfer from Lisp to R was very great). Watching inexperienced students coping with Lisp-Stat and R reinforces this judgment about their relative difficulty. Moreover, a number of the members of R Core made similar points. One said, for example, "I think Lisp-Stat is great, I like it a lot, ...[but] I know a fair amount of people, and you probably do too, who just can't relate to the Lisp way of expressing things."

a large and growing user community and developer base. Indeed, much of the dynamism of R is now in package development. The visibility of R has grown, with, for example, the publication of many books that reference R directly or indirectly. As a consequence, quite a few individuals and organizations now have a substantial investment in R. I have already noted the recent *New York Times* article on R, and the fact that commercial statistical-software developers have found it advantageous to try to integrate R in their products.

Nevertheless, several aspects of the social organization of the R Project pose challenges to its continued success.

Decision making and division of labour in R Core

The decision-making procedures of the R Core team were perhaps better suited to an earlier stage in the development of the software and to a smaller Core group. At an earlier phase in the development of R, so much work needed to be done that direction and coordination were less critical issues than they now appear to be. Moreover, the modified-consensus decision-making procedure of the R Core team and its dependence on voluntarism have apparently prevented some long-standing issues from being adequately addressed.

Several members of R Core mentioned this problem to me—although they don't necessarily agree on what the outstanding issues are. For example,

> There are several major issues which have been around for a long time. One major issue is how to, how closely to integrate S4 with "Base R" .... And then performance is still a major issue ... [We] are gradually moving towards isolating the problems and working on the obstacles, gradually eliminating them. But there's no sort of concerted effort in the sense that we say, "OK, let's all sit together and figure out what the actions are that need to be taken, and let's do an action plan and all that stuff." Development doesn't work along these lines.

With respect to R infrastructure for handling large data sets, an issue that came up in several interviews, one member of the R Core team remarked,

> I think for a lot of the stuff, if you look at the R development, it happens when someone has the need for it. If it hasn't happened thus far, it's because all the 17 core developers haven't had the specific need for that kind of thing, but eventually it will happen.

Two observations about this last remark: (1) As I explained earlier, a great deal of the activity of members of R Core (e.g., the implementation of internationalization) is not the product of direct need, and so it would clearly be wrong to conclude the the contents of the base R system simply reflect the needs of the core developers. (2) There are indeed facilities in R for handling large quantities of data, as reflected, for example, in various interfaces to database-management systems, R packages for handling massive genomic data sets, and packages, such as the **biglm** (Lumley, 2009a) and **survey** (Lumley, 2009b) packages, that are specifically designed to deal with large problems. Nevertheless, I believe that the basic point is a valid one: At least in naive use, R users often encounter memory issues, as evidenced by the frequency with which such issues are raised on the r-help email list.

Similarly, another member of R Core remarked,

> I have more ambitious plans for package mechanisms, and threading is looming large on the horizon all the time, but there's a lot of resistance to making major changes, and that inhibits me from doing a lot of stuff there.

He attributed the resistance to two factors: a "need to maintain the user base," and the lack of time among members of R Core to do "long term scheduling."

There is another side to the coin of modified-consensus decision-making that has been pointed out to me as well: Requiring consensus, or at least acquiesence, avoids the implementation of half-baked solutions to problems.

I have already noted the apparently unequal division of labour among the members of the R Core group. There is, therefore, a potential over-dependence upon a few key individuals, and no clear plan for succession if these individuals were for some reason to drop out or seriously to curtail their activities. As one member of R Core put it to me, "[T]here are several small groups of people who, if they fairly simultaneously dropped out, I suspect that it would be very difficult to keep the R Project running."

Tension between innovation and backwards compatibility

An advantage that the R Core team has relative to commercial software developers is that they are not as tightly constrained by their "customers." For example, commercial developers risk alienating their customers if they fail to maintain strict backwards compatibility of their software. Such constraints on R are not entirely absent, however, and several members of R Core (one of whom was quoted above) mentioned them to me in the interviews that I con-

ducted. Speaking of partial name matching, for example, one R Core member told me:

> I think that was a bad design decision. I don't think that's ever going to change, though. I think it's too widely used, even though it should change, and ... the pain would be worth the gain, ... [but] it won't happen.

And more generally,

> I don't think [R] will change radically. I would like it to change, I would like to introduce new capabilities, ... but I think it's too big now to have those accepted.

Negotiating CRAN

The current organization of CRAN — essentially a flat, alphabetized list of nearly 2000 packages — may not be sustainable. It is already difficult for users to navigate the package archive to find resources within it. The problem, however, isn't inherent to CRAN itself, but rather suggests the provision of tools to negotiate the contents of the package archive. Some at least partly successful solutions have already been implemented:

Search Tools There are several search mechanisms, both inside and outside of R, that help users to locate resources — for example, the `help.search` and `RSiteSearch` commands in R; various search sites, such as `http://www.rseek.org/`; the **sos** package (Graves et al., 2009); and the *Crantastic* web site, `http://crantastic.org/`, which has both package search and tagging features. My experience with these facilities is that they are useful, but they often both produce large numbers of irrelevant hits and miss relevant information.

Task Views The CRAN Task Views (Zeileis, 2005) are a serious attempt to help users to navigate the package system. In my opinion, the task views are valuable, but fall substantially short of providing complete and convenient access to the contents of CRAN. As I can attest from personal experience, it is difficult, for example, for maintainers of task views to keep them current. Nor are task views *intended* to provide a complete index to CRAN.

Keywords Help files include mandatory keywords that could, in principle, provide a basis for searching CRAN, but the standard set of keywords is not very helpful. Similarly, package 'DESCRIPTION' files have an optional `Classification` field that can be used to tag the package's content according to several standard schemes, but this field is little used by package authors. Similarly, R `.Rd` files support optional `\concept` markup, which allows a package writer to supply arbitrary keywords, accessible through the `help.search` command. This facility too is mostly ignored. In contrast, the Bioconductor package archive takes a more sophisticated and prescriptive approach to the use of keywords, which are supplied via a mandatory `biocViews` field to the package 'DESCRIPTION' file. The keywords are then used to subset packages.

It is hard to imagine that, without further development, the current structure of CRAN and the tools that surround it could usefully survive, say, five more years of exponential growth. Paradoxically, then, R is challenged by its own success. That said, providing more effective mechanisms for organizing the information in CRAN is a difficult problem: I don't mean either to minimize the difficulty of the problem or to disparage the efforts that have been directed at it thus far.

Concluding remarks

A wild card in the future of R is the possible development of competing statistical software that breaks radically with the structure of the S language. It is easier, however, to understand some of the social structures and processes that contribute to R's current momentum and others that present challenges to its continued development. The sum of these more predictable factors suggests that at least the short-term prospects for the R Project remain bright. But, as the American baseball player Yogi Berra famously said, "It ain't over till it's over."

Acknowledgments

The research reported in this paper was supported by grants from the McMaster University Arts Research Board and the Social Sciences and Humanities Research Council of Canada. The paper is partly based on a presentation that I gave at the useR! 2008 conference in Dortmund, Germany. I am grateful to the members of R Core and others associated with the R Project who consented to be interviewed; to Bonnie Fox for comments on an early draft of this paper; and to two reviewers and the editor of The R Journal, whose critical comments on a first version of the paper stimulated various revisions. Any remaining deficiencies are, of course, my responsibility.

Bibliography

J. W. Eaton, D. Bateman, and S. Hauberg. *GNU Octave*. Free Software Foundation, Boston, third edition, 2008.

K. Fogel. *Producing Open Source Software*. O'Reilly, Sebastopol CA, 2006.

J. Fox. *An R and S-PLUS Companion to Applied Regression*. Sage, Thousand Oaks CA, 2002.

J. Fox. Editorial. *R News*, 8(2):1–2, 2008.

S. Graves, S. Dorai-Raj, and R. Francois. *sos*, 2009. R package version 1.1-3.

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

D. Knoke. Associations and interest groups. *Annual Review of Sociology*, 12:1–21, 1986.

F. Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, 2002.

F. Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, 2003.

T. Lumley. *biglm: bounded memory linear and generalized linear models*, 2009a. R package version 0.7.

T. Lumley. *survey: analysis of complex survey samples*, 2009b. R package version 3.16.

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and the R Core team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2009. R package version 3.1-94.

R. D. Putnam. Bowling alone: America's declining social capital. *The Journal of Democracy*, 6(1):65–78, 1995.

E. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, Sebastopol CA, 2001a.

E. Raymond. The cathedral and the bazaar:. In *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, pages 19–64. O'Reilly, Sebastopol CA, 2001b.

E. Raymond. Homesteading the noosphere. In *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, pages 65–112. O'Reilly, Sebastopol CA, 2001c.

E. Raymond. The magic cauldron. In *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, pages 113–166. O'Reilly, Sebastopol CA, 2001d.

D. H. Smith. Voluntary action and voluntary groups. *Annual Review of Sociology*, 1:247–270, 1975.

R. M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, 2002.

T. Therneau and T. Lumley. *survival: Survival analysis, including penalised likelihood*, 2009. R package version 2.35-7.

L. Tierney. *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. Wiley, New York, 1990.

A. Vance. Data analysts captivated by R's power. *The New York Times*, 158(54548), 2009. URL http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=2&pagewanted=1.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.

S. Weber. *The Success of Open Source*. Harvard University Press, Cambridge MA, 2004.

S. Winchester. *The Meaning of Everything: The Story of the Oxford English Dictionary*. Oxford University Press, Oxford, 2003.

A. Zeileis. CRAN task views. *R News*, 5(1):39–40, 2005.

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

Party on!

A New, Conditional Variable-Importance Measure for Random Forests Available in the party Package

by Carolin Strobl, Torsten Hothorn and Achim Zeileis

Abstract: Random forests are one of the most popular statistical learning algorithms, and a variety of methods for fitting random forests and related recursive partitioning approaches is available in R. This paper points out two important features of the random forest implementation `cforest` available in the **party** package: The resulting forests are unbiased and thus preferable to the `randomForest` implementation available in **randomForest** if predictor variables are of different types. Moreover, a conditional permutation importance measure has recently been added to the **party** package, which can help evaluate the importance of correlated predictor variables. The rationale of this new measure is illustrated and hands-on advice is given for the usage of recursive partitioning tools in R.

Recursive partitioning methods are amongst the most popular and widely used statistical learning tools for nonparametric regression and classification. Random forests in particular, which can deal with large numbers of predictor variables even in the presence of complex interactions, are being applied successfully in many scientific fields (see, e.g., Lunetta et al., 2004; Strobl et al., 2009, and the references therein for applications in genetics and social sciences). Thus, it is not surprising that there is a variety of recursive partitioning tools available in R (see `http://CRAN.R-project.org/view=MachineLearning` for an overview).

The scope of recursive partitioning methods in R ranges from the standard classification and regression trees available in **rpart** (Therneau et al., 2008) to the reference implementation of random forests (Breiman, 2001) available in **randomForest** (Liaw and Wiener, 2002, 2008). Both methods are popular in applied research, and several extensions and refinements have been suggested in the statistical literature in recent years.

One particularly important improvement was the introduction of unbiased tree algorithms, which overcome the major weak spot of the classical approaches available in **rpart** and **randomForest**: variable-selection bias. The term *variable-selection bias* refers to the fact that in standard tree algorithms variable selection is biased in favor of variables offering many potential cut-points, so that variables with many categories and continuous variables are artificially preferred (see, e.g, Kim and Loh, 2001; Shih, 2002; Hothorn et al., 2006; Strobl et al., 2007a, for details).

To overcome this weakness of the early tree algorithms, new algorithms have been developed that do not artificially favor splits in variables with many categories or continuous variables. In R such an unbiased tree algorithm is available in the `ctree` function for conditional inference trees in the **party** package (Hothorn et al., 2006). The package also provides a random forest implementation `cforest` based on unbiased trees, which enables learning unbiased forests (Strobl et al., 2007b).

Unbiased variable selection is the key to reliable prediction and interpretability in both individual trees and forests. However, while a single tree's interpretation is straightforward, in random forests an extra effort is necessary to assess the importance of each predictor in the complex ensemble of trees.

This issue is typically addressed by means of variable-importance measures such as Gini importance and the "mean decrease in accuracy" or "permutation" importance, available in **randomForest** in the `importance()` function (with `type = 2` and `type = 1`, respectively). Similarly, a permutation-importance measure for `cforest` is available via `varimp()` in **party**.

Unfortunately, variable-importance measures in random forests are subject to the same bias in favor of variables with many categories and continuous variables that affects variable selection in single trees, and also to a new source of bias induced by the resampling scheme (Strobl et al., 2007b). Both problems can be addressed in **party** to guarantee unbiased variable selection and variable importance for predictor variables of different types.

Even though this refined approach can provide reliable variable-importance measures in many applications, the original permutation importance can be misleading in the case of correlated predictors. Therefore, Strobl et al. (2008) suggested a solution for this problem in the form of a new, conditional permutation-importance measure. Starting from version 0.9-994, this new measure is available in the **party** package.

The rationale and usage of this new measure is outlined in the following sections and illustrated by means of a toy example.

Random forest variable-importance measures

Permutation importance, which is available in **randomForest** and **party**, is based on a random permutation of the predictor variables, as described in more detail below.

The alternative variable-importance measure

available in **randomForest**, Gini importance, is based on the Gini gain criterion employed in most traditional classification tree algorithms. However, the Gini importance has been shown to carry forward the bias of the underlying Gini-gain splitting criterion (see, e.g., Kim and Loh, 2001; Strobl et al., 2007a; Hothorn et al., 2006) when predictor variables vary in their number of categories or scale of measurement (Strobl et al., 2007b). Therefore, it is not recommended in these situations.

Permutation importance, on the other hand, is a reliable measure of variable importance for uncorrelated predictors when sub-sampling without replacement — instead of bootstrap sampling — and unbiased trees are used in the construction of the forest (Strobl et al., 2007b). Accordingly, the default settings for the control parameters `cforest_control` have been pre-defined to the default version `cforest_unbiased` to guarantee subsampling without replacement and unbiased individual trees in fitting random forests with the **party** package.

The rationale of the original permutation-importance measure is the following: By randomly permuting the predictor variable X_j, its original association with the response Y is broken. When the permuted variable X_j, together with the remaining non-permuted predictor variables, is used to predict the response for the out-of-bag observations, the prediction accuracy (i.e., the number of correctly classified observations in classification, or respectively the mean squared error in regression) decreases substantially if the original variable X_j was associated with the response. Thus, Breiman (2001) suggests the difference in prediction accuracy before and after permuting X_j, averaged over all trees, as a measure for variable importance.

In standard implementations of random forests, such as **randomForest** in R, an additional scaled version of permutation importance (often called the z-score), which is computed by dividing the raw importance by its standard error, is provided (for example by `importance(obj, type = 2, scale = TRUE)` in **randomForest**). Note, however, that the results of Strobl and Zeileis (2008) show that the z-score is not suitable for significance tests and that raw importance has better statistical properties.

Why conditional importance?

The original permutation-importance measure can, for reasons outlined below, be considered as a marginal measure of importance. In this sense, it has the same property as, e.g., a marginal correlation coefficient: A variable that has no effect of its own, but is correlated with a relevant predictor variable, can receive a high importance score. Accordingly, empirical results (see Strobl et al., 2008, and the references therein) suggest that the original permutation-importance measure often assigns higher scores to correlated predictors.

In contrast, partial correlation coefficients (like the coefficients in linear regression models) measure the importance of a variable given the other predictor variables in the model. The advantage of such a partial, or conditional, approach is illustrated by means of a toy example: The data set `readingSkills` is an artificial data set generated by means of a linear model. The response variable contains hypothetical scores on a test of reading skills for 200 school children. Potential predictor variables in the data set are the age of the child, whether the child is a native speaker of the test language and the shoe size of the child.

Obviously, the latter is not a sensible predictor of reading skills (and was actually simulated not to have any effect on the response) — but with respect to marginal (as opposed to partial) correlations, shoe size is highly correlated with the test score. Of course this spurious correlation is only due to the fact that both shoe size and test score are associated with the underlying variable age.

In this simple problem, a linear model would be perfectly capable of identifying the original coefficients of the predictor variables (including the fact that shoe size has no effect on reading skills once the truly relevant predictor variable age is included in the model). However, the `cforest` permutation-importance measure is mislead by the spurious correlation and assigns a rather high importance value to the nonsense-variable shoe size:

```
> library("party")
> set.seed(42)
> readingSkills.cf <- cforest(score ~ .,
+    data = readingSkills, control =
+    cforest_unbiased(mtry = 2, ntree = 50))

> set.seed(42)
> varimp(readingSkills.cf)

nativeSpeaker           age      shoeSize
     12.62036      74.52034      17.97287
```

The reason for this odd behavior can be found in the way the predictor variables are permuted in the computation of the importance measure: Strobl et al. (2008) show that the original approach, where one predictor variable X_j is permuted against both the response Y and the remaining (one or more) predictor variables $Z = X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_p$, as illustrated in Figure 1, corresponds to a pattern of independence between X_j and both Y and Z.

From a theoretical point of view, this means that a high value of the importance measure can be caused by a violation either of the independence between X_j and Y or of the independence between X_j and Z, even though the latter is not of interest here. For practical applications, this means that a variable X_j

that is correlated with an important predictor Z can appear more important than an uncorrelated variable, even if X_j has no effect of its own.

Y	X_j	Z
y_1	$x_{\pi_j(1),j}$	z_1
$\vdots$	$\vdots$	$\vdots$
y_i	$x_{\pi_j(i),j}$	z_i
$\vdots$	$\vdots$	$\vdots$
y_n	$x_{\pi_j(n),j}$	z_n

Figure 1: Permutation scheme for the original permutation-importance measure.

Y	X_j	Z
y_1	$x_{\pi_{j\|Z=a}(1),j}$	$z_1 = a$
y_3	$x_{\pi_{j\|Z=a}(3),j}$	$z_3 = a$
y_{27}	$x_{\pi_{j\|Z=a}(27),j}$	$z_{27} = a$
y_6	$x_{\pi_{j\|Z=b}(6),j}$	$z_6 = b$
y_{14}	$x_{\pi_{j\|Z=b}(14),j}$	$z_{14} = b$
y_{21}	$x_{\pi_{j\|Z=b}(21),j}$	$z_{21} = b$
$\vdots$	$\vdots$	$\vdots$

Figure 2: Permutation scheme for the conditional permutation importance.

The aim to reflect only the impact of X_j itself in predicting the response Y, rather than its correlations with other predictor variables, can be better achieved by means of a conditional importance measure in the spirit of a partial correlation: We want to measure the association between X_j and Y given the correlation structure between X_j and the other predictor variables in the data set.

To meet this aim, Strobl et al. (2008) suggest a conditional permutation scheme, where X_j is permuted only within groups of observations with $Z = z$ in order to preserve the correlation structure between X_j and the other predictor variables, as illustrated in Figure 2.

With this new, conditional permutation scheme, the importance measure is able to reveal the spurious correlation between shoe size and reading skills:

```
> set.seed(42)
> varimp(readingSkills.cf, conditional =
+     TRUE)

nativeSpeaker           age      shoeSize
    11.161887     44.388450      1.087162
```

Only by means of conditional importance it becomes clear that the covariate native speaker is actually more relevant for predicting the test score than shoe size is, whose conditional effect is negligible. Thus, the conditional importance mimics the behavior of partial correlations or linear regression coefficients, the interpretation of which many readers may be more familiar. However, whether a marginal or conditional importance measure is to be preferred depends on the actual research question.

Current research also investigates the impact that the choice of the tuning parameter `mtry`, which regulates the number of randomly preselected predictor variables that can be chosen in each split (cf. Strobl et al., 2008), and parameters regulating the depth of the trees have on variable importance.

How is the conditioning grid defined technically?

Conditioning is straightforward whenever the variables to be conditioned on, Z, are categorical (cf., e.g., Nason et al., 2004). However, conditioning on continuous variables, which may entail as many different values as observations in the sample, would produce cells with very sparse counts — which would make permuting the values of X_j within each cell rather pointless. Thus, in order to create cells of reasonable size for conditioning, continuous variables need to be discretized.

As a straightforward discretization strategy for random forests, Strobl et al. (2008) suggest defining the conditioning grid by means of the partition of the feature space induced by each individual tree. This grid can be used to conditionally permute the values of X_j within cells defined by combinations of Z, where Z can contain potentially large sets of covariates of different scales of measurement.

The main advantages of this approach are that this partition has already been learned from the data during model fitting, that it can contain splits in categorical, ordered and continuous predictor variables, and that it can thus serve as an internally available means for discretizing the feature space. For ease of computation, the conditioning grid employed in `varimp` uses all cut-points as bisectors of the sample space (the same approach is followed by Nason et al., 2004).

The set of variables Z to be conditioned on should contain all variables that are correlated with the current variable of interest X_j. In the `varimp` function, this is assured by the small default value 0.2 of the `threshold` argument: By default, all variables whose correlation with X_j meets the condition $1 - p-\text{value} > 0.2$ are used for conditioning. A larger value of `threshold` would have the effect that only those variables that are strongly correlated with X_j would be used for conditioning, but would also lower the computational burden.

Note that the same permutation tests that are used for split selection in the tree building process (Hothorn et al., 2006) are used here to measure the association between X_j and the remaining covariates.

A short recipe for fitting random forests and computing variable importance measures with R

To conclude, we would like to summarize the application of conditional variable importance and general issues in fitting random forests with R. Depending on certain characteristics of your data set, we suggest the following approaches:

- If all predictor variables are of the same type (for example: all continuous or all unordered categorical with the same number of categories), use either `randomForest` (**randomForest**) or `cforest` (**party**). While `randomForest` is computationally faster, `cforest` is safe even for variables of different types.

  For predictor variables of the same type, Gini importance, `importance(obj, type = 2)`, or permutation importance, `importance(obj, type = 1)`, available for `randomForest`, and permutation importance, `varimp(obj)`, available for `cforest`, are all adequate importance measures.

- If the predictor variables are of different types (for example: different scales of measurement, different numbers of categories), use `cforest` (**party**) with the default option `controls = cforest_unbiased` and permutation importance `varimp(obj)`.

- If the predictor variables are correlated, depending on your research question, conditional importance, available via `varimp(obj, conditional = TRUE)` for `cforest` (**party**), can add to the understanding of your data.

General remarks:

- Note that the default settings for `mtry` differ in `randomForest` and `cforest`: In `randomForest` the default setting for classification, e.g., is `floor(sqrt(ncol(x)))`, while in `cforest` it is fixed to the value 5 for technical reasons.

- Always check whether you get the same results with a different random seed before interpreting the variable importance ranking!

  If the ranking of even the top-scoring predictor variables depends on the choice of the random seed, increase the number of trees (argument `ntree` in `randomForest` and `cforest_control`).

Bibliography

L. Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

H. Kim and W. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454):589–604, 2001.

A. Liaw and M. Wiener. Classification and regression by `randomForest`. *R News*, 2(3):18–22, 2002.

A. Liaw and M. Wiener. *`randomForest`: Breiman and Cutler's Random Forests for Classification and Regression*, 2008. URL `http://CRAN.R-project.org/package=randomForest`. R package version 4.5-28.

K. L. Lunetta, L. B. Hayward, J. Segal, and P. V. Eerdewegh. Screening large-scale association study data: Exploiting interactions using random forests. *BMC Genetics*, 5:32, 2004.

M. Nason, S. Emerson, and M. Leblanc. CARTscans: A tool for visualizing complex models. *Journal of Computational and Graphical Statistics*, 13(4):1–19, 2004.

Y.-S. Shih. Regression trees with unbiased variable selection. *Statistica Sinica*, 12:361–386, 2002.

C. Strobl and A. Zeileis. Danger: High power! – Exploring the statistical properties of a test for random forest variable importance. In *Proceedings of the 18th International Conference on Computational Statistics, Porto, Portugal*, 2008.

C. Strobl, A.-L. Boulesteix, and T. Augustin. Unbiased split selection for classification trees based on the Gini index. *Computational Statistics & Data Analysis*, 52(1):483–501, 2007a.

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8:25, 2007b.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9:307, 2008.

C. Strobl, J. Malley, and G. Tutz. An introduction to recursive partitioning: Rationale, application and characteristics of classification and regression trees, bagging and random forests. *Psychological Methods*, 2009. In press.

T. M. Therneau, B. Atkinson, and B. D. Ripley. `rpart`: Recursive partitioning. 2008. URL `http://CRAN.R-project.org/package=rpart`. R package version 3.1-41.

Carolin Strobl
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
`carolin.strobl@stat.uni-muenchen.de`

ConvergenceConcepts: An R Package to Investigate Various Modes of Convergence

by Pierre Lafaye de Micheaux and Benoit Liquet

Abstract: **ConvergenceConcepts** is an R package, built upon the **tkrplot**, **tcltk** and **lattice** packages, designed to investigate the convergence of simulated sequences of random variables. Four classical modes of convergence may be studied, namely: almost sure convergence (*a.s.*), convergence in probability (*P*), convergence in law (*L*) and convergence in *r*-th mean (*r*). This investigation is performed through accurate graphical representations. This package may be used as a pedagogical tool. It may give students a better understanding of these notions and help them to visualize these difficult theoretical concepts. Moreover, some scholars could gain some insight into the behaviour of some random sequences they are interested in.

Introduction

Many students are exposed, during their graduate school years, to the difficult concepts of convergence of a sequence of random variables (see Sethuraman (1995)). Indeed, as pointed out by Bryce, Gould, Notz and Peck (2001), "statistical theory is an important part of the curriculum, and is particularly important for students headed for graduate school". Such knowledge is prescribed by learned statistics societies (see Accreditation of Statisticians by the Statistical Society of Canada, and Curriculum Guidelines for Undergraduate Programs in Statistical Science by the American Statistical Association). In the main textbooks (see for example Billingsley (1986), Chung (1974), Ferguson (1996), Lehmann (2001), Serfling (2002)), around 15 pages without graphs are allotted to defining these convergence concepts and their interrelations. But, very often, these concepts are only described through their definitions and some of their properties. Thus, some students may not fully visualize how a random variable converges to some limit. They also may not fully understand the differences between the various modes, especially between convergence in probability and almost surely.

Moreover, a statistician could be interested in whether or not a specific random sequence converges. To explain the modes of convergence, we could follow Bryce, Gould, Notz and Peck (2001)'s advice: "a modern statistical theory course might, for example, include more work on computer intensive methods". With regard to the convergence in law, Dunn (1999) and Marasinghe, Meeker, Cook and Shin (1996) have proposed tools to explain this concept in an interactive manner. Mills (2002) proposed a review of statistical teaching based on simulation methods and Chance and Rossman (2006) have written a book on this subject. Our package enables one to investigate graphically the four classical modes of convergence of a sequence of random variables: convergence almost surely, convergence in probability, convergence in law and convergence in *r*-th mean. Note that it is tightly associated with the reading of Lafaye de Micheaux and Liquet (2009) which explains what we call our "mind visualization approach" of these convergence concepts.

The two main functions to use in our package are `investigate` and `check.convergence`. The first one will be described in the next section, investigating pre-defined Exercise 1 from Lafaye de Micheaux and Liquet (2009). The second one will be described in the last section, where it is shown how to treat your own examples.

At this point, note the necessary first two steps to perform before working with our package:

```
install.packages("ConvergenceConcepts")
require(ConvergenceConcepts)
```

Pre-defined examples

Our package contains several pre-defined examples and exercises (all introduced and solved in Lafaye de Micheaux and Liquet (2009) and in its associated online Appendix), some of which are classical ones. To investigate these examples, just type in the R console:

```
investigate()
```

Any entry can be selected by clicking in the left panel displayed in Figure 1. The corresponding text then appears inside the right panel. Next, by clicking the OK button, the relevant R functions are called to help the user to visualize the chosen modes of convergence for the random variable sequence under investigation. You will then be able to twiddle a few features.

For example, the first entry corresponds to the following problem.

Exercise 1: Let Z be a uniform $U[0,1]$ random variable and define $X_n = 1_{[m/2^k,(m+1)/2^k]}(Z)$ where $n = 2^k + m$ for $k \geq 1$ and with $0 \leq m < 2^k$. Thus $X_1 = 1$, $X_2 = 1_{[0,1/2)}(Z)$, $X_3 = 1_{[1/2,1)}(Z)$, $X_4 = 1_{[0,1/4)}(Z)$, $X_5 = 1_{[1/4,1/2)}(Z)$, ... Does $X_n \xrightarrow{a.s.} 0$? Does $X_n \xrightarrow{P} 0$?

Solution to exercise 1: The drawing on Figure 2 explains the construction of X_n.

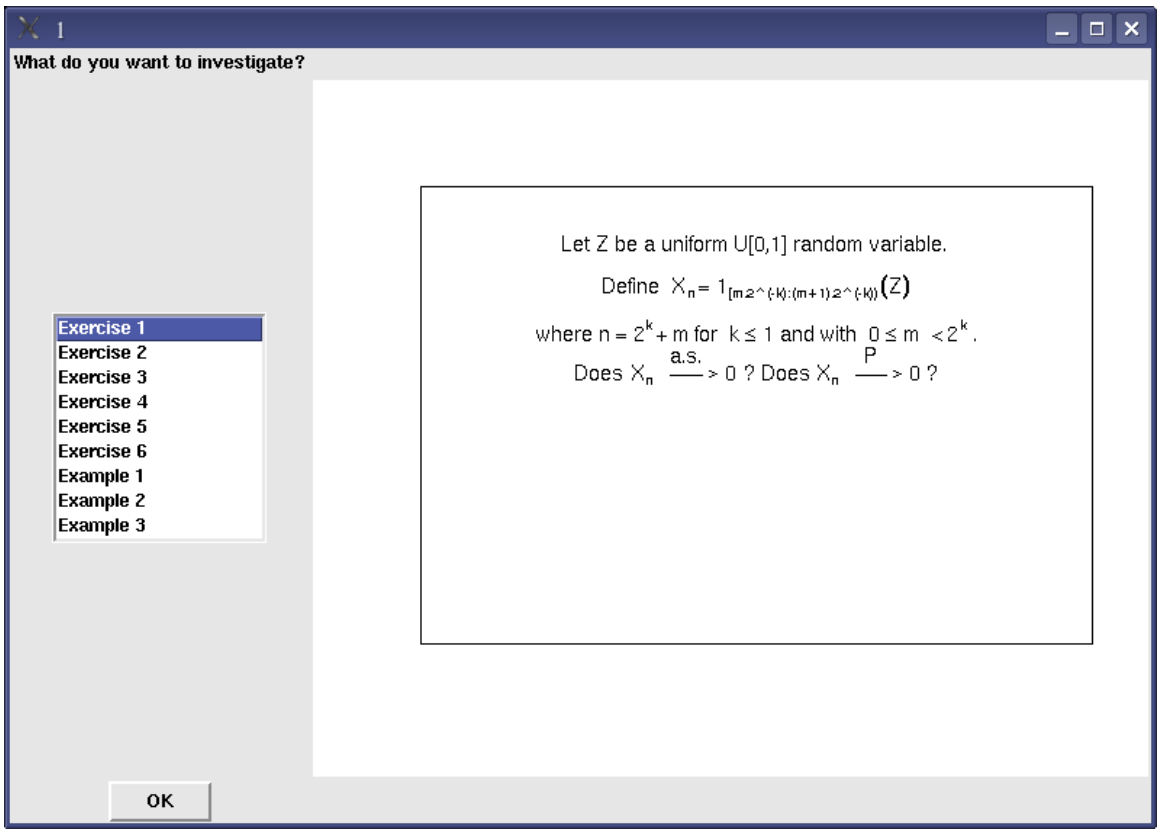

Figure 1: A call to `investigate()`.

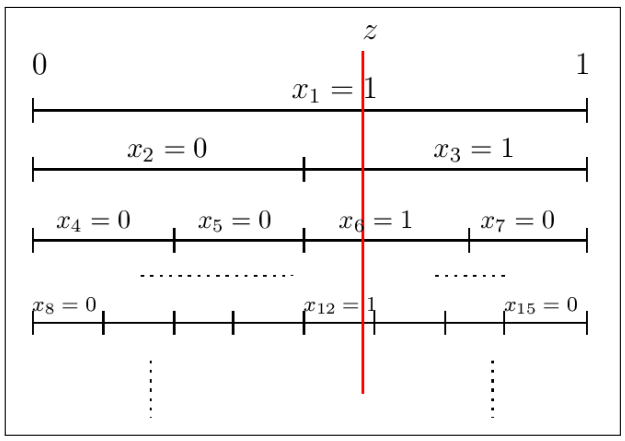

Figure 2: One fictitious sample path for X_n.

Let us apply our visual reasoning as explained in Section 2 of (Lafaye de Micheaux and Liquet, 2009). Once a z value is randomly drawn, the entire associated sample path is fully determined. As n increases, each sample path "stays" for a longer time at 0 but eventually jumps to 1. In fact it will jump to 1 an infinite number of times after each fixed n value.

This can be seen using our package. After having selected the first entry and clicked on the OK button displayed in Figure 1, we obtain the following plot.

On the left panel of this plot, we see some sample paths associated with X_n. One can thus notice, on left panel of Figure 3, that for all $n = 1, \ldots$, all these sample paths will jump to 1 somewhere (and even at many places) in the grey block beginning at position n.

By definition, $X_n \xrightarrow{a.s.} 0$ if and only if $\forall \epsilon > 0 : a_n = \mathbb{P}[\omega; \exists k \geq n : |X_{k,\omega}| > \epsilon] \underset{n\to\infty}{\longrightarrow} 0$. In this definition, and in the one thereafter on convergence in probability, ω can be viewed as some kind of labelling of a sample path. We define $\hat{a}_n$ to be a frequentist estimate of a_n. It is the proportion of the pieces of generated sample paths beginning at position n that go outside the horizontal band $[-\epsilon, +\epsilon]$ (see Lafaye de Micheaux and Liquet (2009) for more details). Using our package, the user can interactively move the grey block on left side of Figure 3. He can thus observe the pieces of sample paths which leave the horizontal band. Red marks indicate, for each sample path, the first time when this happens. Simultaneously we can observe their proportion $\hat{a}_n$ (equal to 1 here) on right side of Figure 3 as indicated by a sliding red circle. We can see that we cannot have almost sure convergence.

By definition, $X_n \xrightarrow{P} 0$ if and only if $\forall \epsilon > 0 : \; p_n = \mathbb{P}[\omega; \; |X_{n,\omega}| > \epsilon] \underset{n\to\infty}{\longrightarrow} 0$. We define $\hat{p}_n$ to be a frequentist estimate of p_n. It is the proportion of generated sample paths lying outside a band $[-\epsilon, +\epsilon]$ in the bar at position n. Note that, for this example, this corresponds to the proportion of $[0,1]$-uniform z values falling into an interval whose length gets narrower. We can investigate graphically convergence in prob-

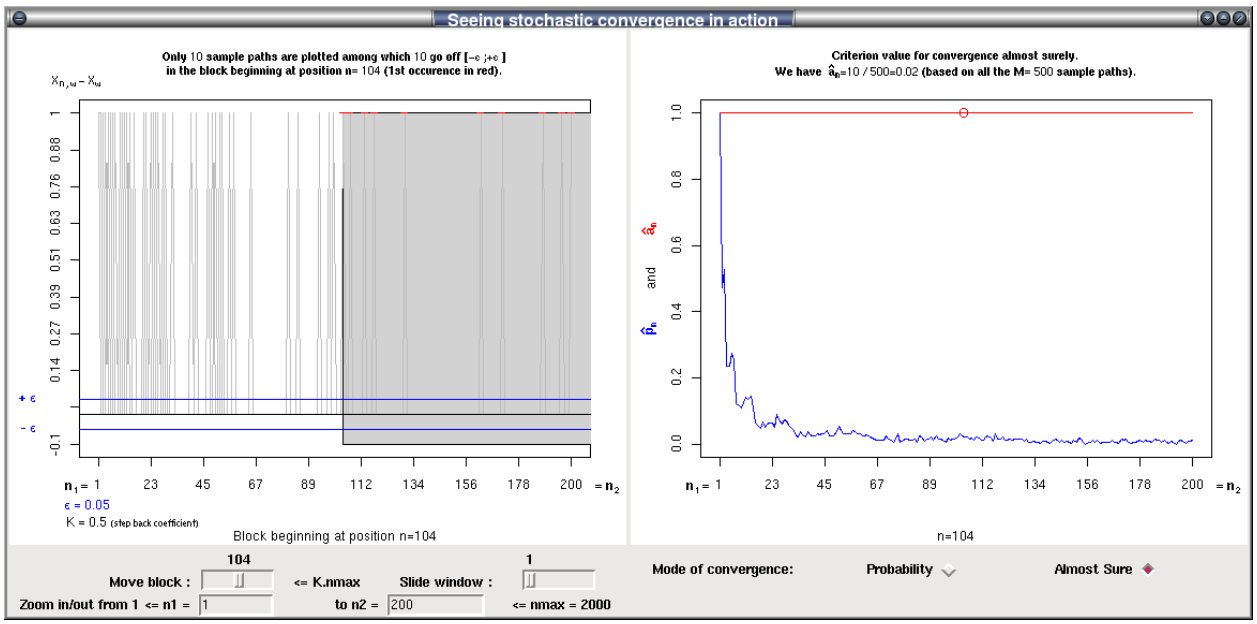

Figure 3: Seeing almost sure convergence in action. On the left panel, we visualize only 10 sample paths among the $M = 500$ simulated realizations of the sequence of random variables defined in exercise 1. It can be seen that all these sample paths go off $[-\epsilon, +\epsilon]$ in the block beginning at position $n = 104$. On the right panel, we visualize the $\hat{a}_n$ values for each n between 1 and 200. We see that $\hat{a}_n$ equal 1.

ability by sliding the vertical bar (click first on radio button *Probability* for this bar to appear, see Figure 4) and observe that $\hat{p}_n$ is going towards 0. This lets us perceive that in this case, we do have convergence in probability.

For the interested reader, a mathematically rigorous proof of this exercise can be found in Lafaye de Micheaux and Liquet (2009).

Investigating your own examples

We now introduce two new problems that have not been either pre-included or treated in the package. We show how a user can define his own functions in order to investigate the convergence of X_n towards X, or equivalently of $X_n - X$ to 0. These problems are rather simple, but the objective here is only to show how to use our package. The two steps will consist in coding your generator of the X_i's and then using the `check.convergence` function.

This last function has several arguments whose description is now given.

nmax: number of points in each sample path.

M: number of sample paths to be generated.

genXn: a function that generates the first n $X_n - X$ values, or only the first n X_n values in the law case.

argsXn: a list of arguments to *genXn*.

mode: a character string specifying the mode of convergence to be investigated, must be one of `"p"` (default), `"as"`, `"r"` or `"L"`.

epsilon: a numeric value giving the interval endpoint.

r: a numeric value (r>0) if convergence in r-th mean is to be studied.

nb.sp: number of sample paths to be drawn on the left plot.

density: if *density*=`TRUE`, then the plot of the density of X and the histogram of X_n is returned. If *density*=`FALSE`, then the plot of the distribution function $F(t)$ of X and the empirical distribution $F_n(t)$ of X_n is returned.

densfunc: function to compute the density of X.

probfunc: function to compute the distribution function of X.

tinf: lower limit for investigating convergence in law.

tsup: upper limit for investigating convergence in law.

trace: function used to draw the plot; `plot` or `points`.

...: optional arguments to *trace*.

Problem 1: Let $X_1, X_2, \ldots$ be independent, identically distributed, continuous random variables with a $N(2,9)$ distribution. Define $Y_i = (0.5)^i X_i$, $i = 1,2,\ldots$. Also define T_n and A_n to be the sum and the average, respectively, of the terms $Y_1, Y_2, \ldots, Y_n$.

(a) Is Y_n convergent in probability to 0?

(b) Is T_n convergent in probability to 2?

(c) Is A_n convergent in probability to 0?

(d) Is T_n convergent in law to a $N(2,3)$?

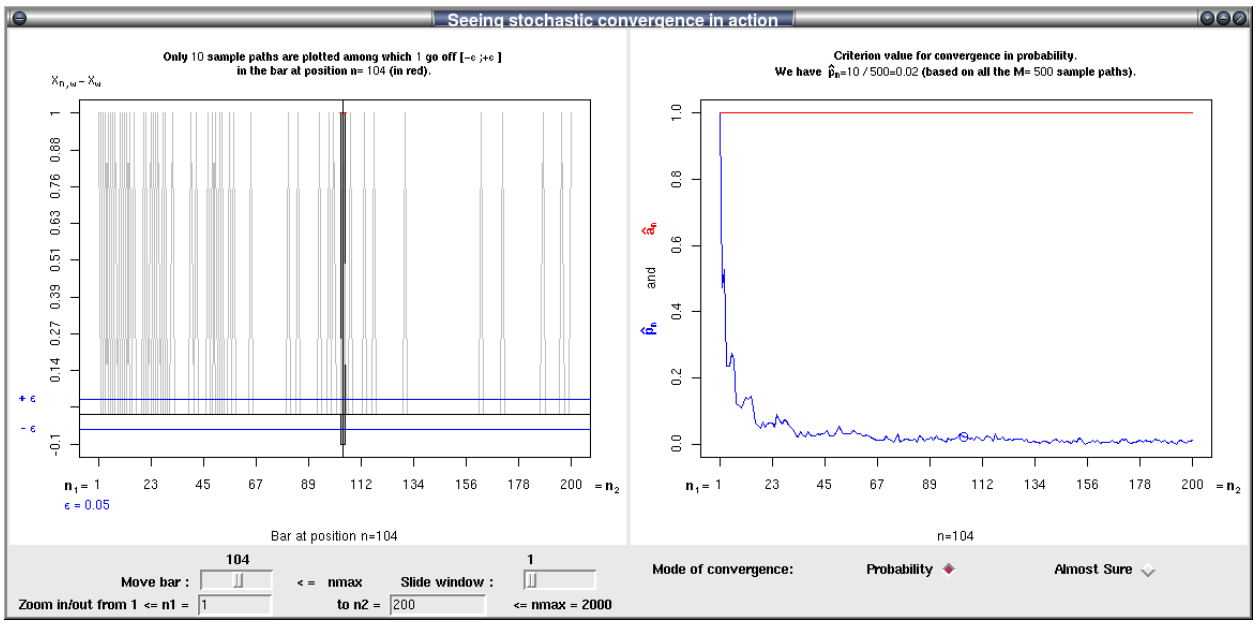

Figure 4: Seeing convergence in probability in action. On the left panel, we visualize only 10 sample paths among the $M = 500$ simulated realizations of the sequence of random variables defined in exercise 1. It can be seen that, among these ten, only one sample path go off $[-\epsilon, +\epsilon]$ in the bar at position $n = 104$. On the right panel, we visualize the $\hat{p}_n$ values for each n between 1 and 200. We see that $\hat{p}_n$ goes towards 0.

Solution to problem 1:

(a) We first define a random generator (called `genYn`) for the Y_i's then we call the function `check.convergence`.

```
genYn <- function(n) {
  res <- (0.5)^(1:n)*rnorm(n,2,3)
  return(res)
}
check.convergence(2000,500,genYn,mode="p")
```

We can zoom in the left panel of Figure 5 (from $n_1 = 1$ to $n_2 = 10$) and see the 10 sample paths going rapidly inside the horizontal band $[-\epsilon, +\epsilon]$. Looking at the evolution of $\hat{p}_n$ towards 0 in the right panel, we can assume that Y_n converges in probability towards 0.

(b) We first define a random generator (called `genTn`) for the $(T_i - 2)$'s then we call the function `check.convergence`.

```
genTn <- function(n) {
  res <- cumsum((0.5)^(1:n)*rnorm(n,2,3))-2
  return(res)
}
check.convergence(2000,500,genTn,mode="p")
```

Each one of the sample paths rapidly evolve towards an horizontal asymptote, not the same for each sample path, and not contained inside the horizontal band $[-\epsilon, +\epsilon]$. Looking at the evolution of $\hat{p}_n$ in the right panel of Figure 6, we can assume that T_n does not converge in probability towards 2.

(c) We first define a random generator (called `genAn`) for the A_i's then we call the function `check.convergence`.

```
genAn <- function(n) {
  x<-1:n
  res<-cumsum((0.5)^x*rnorm(n,2,3))/cumsum(x)
  return(res)
}
check.convergence(2000,500,genAn,mode="p")
```

In this case, we can zoom in (from $n_1 = 1$ to $n_2 = 50$) to better see the sample paths which all end up inside the horizontal band $[-\epsilon, +\epsilon]$. Looking at the evolution of $\hat{p}_n$ towards 0 in the right panel of Figure 7, we can assume that A_n converges in probability towards 0.

(d) We first define a random generator (called `genTnL`) for the T_i's then we call the function `check.convergence`.

```
genTnL <- function(n) {
  res <- cumsum((0.5)^(1:n)*rnorm(n,2,3))
  return(res)
}
check.convergence(2000,1000,genTnL,mode="L",
  density = F,
  densfunc = function(x){dnorm(x,2,sqrt(3))},
  probfunc=function(x){pnorm(x,2,sqrt(3))},
  tinf = -4, tsup = 4)
```

By definition, $X_n \xrightarrow{L} X$ if and only if $l_n(t) = |F_n(t) - F(t)| \underset{n\to\infty}{\longrightarrow} 0$, at all t for which F (the distribution function of X) is continuous, where

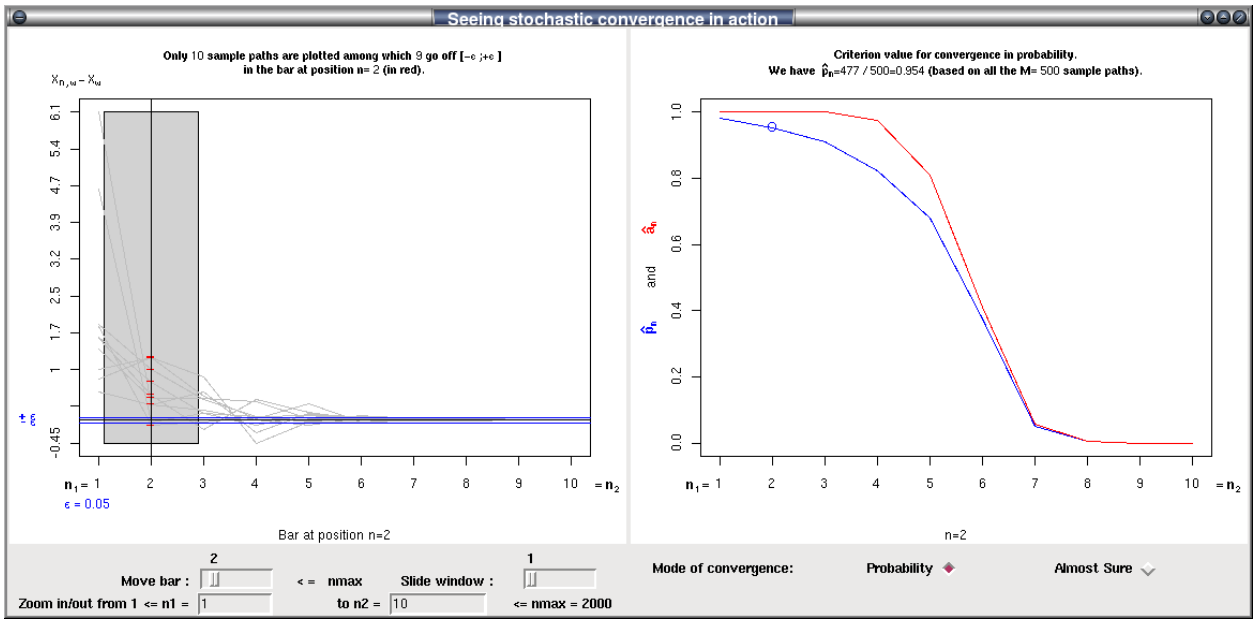

Figure 5: Investigating the convergence in probability towards 0 for the sequence of random variables Y_n of Problem 1.(a). On the right panel, we see that $\hat{p}_n$ goes towards 0.

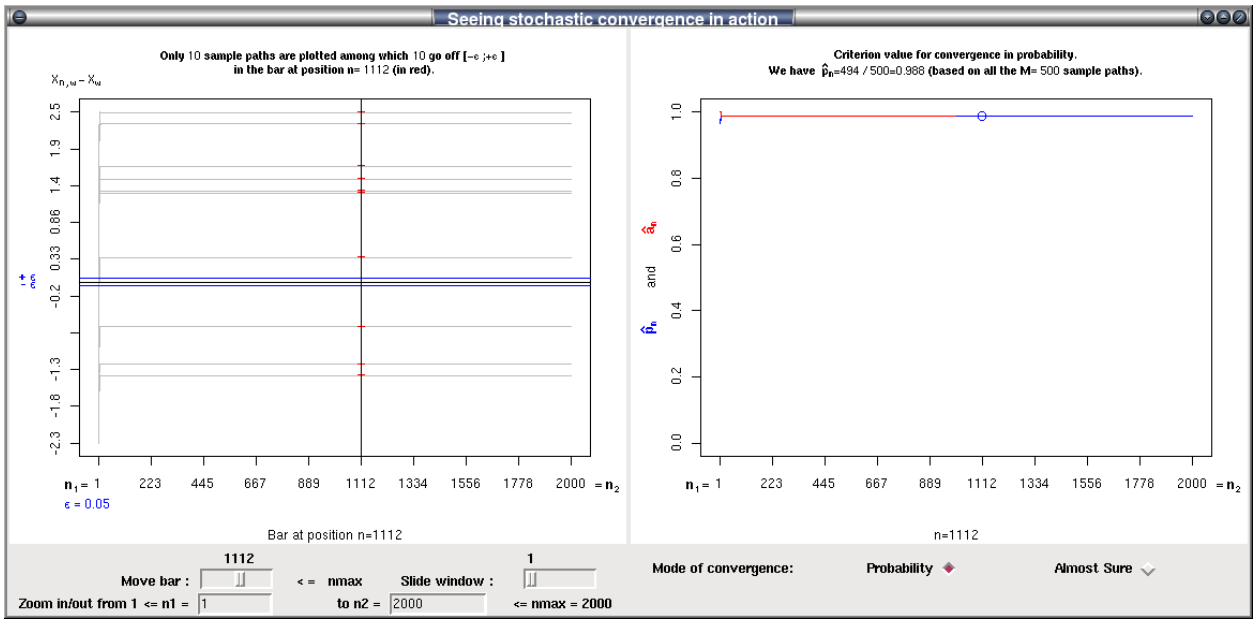

Figure 6: Investigating the convergence in probability towards 2 for the sequence of random variables T_n of Problem 1.(b). On the right panel, we see that $\hat{p}_n$ equals 1.

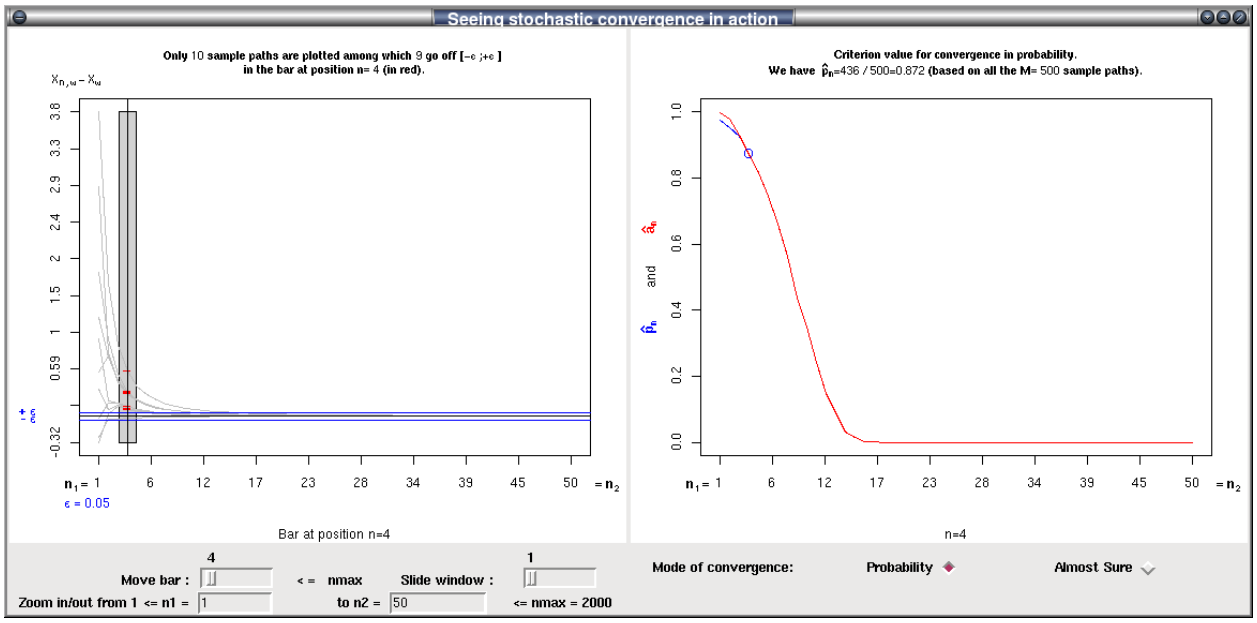

Figure 7: Investigating the convergence in probability towards 0 for the sequence of random variables A_n of Problem 1.(c). On the right panel, we see that $\hat{p}_n$ goes towards 0.

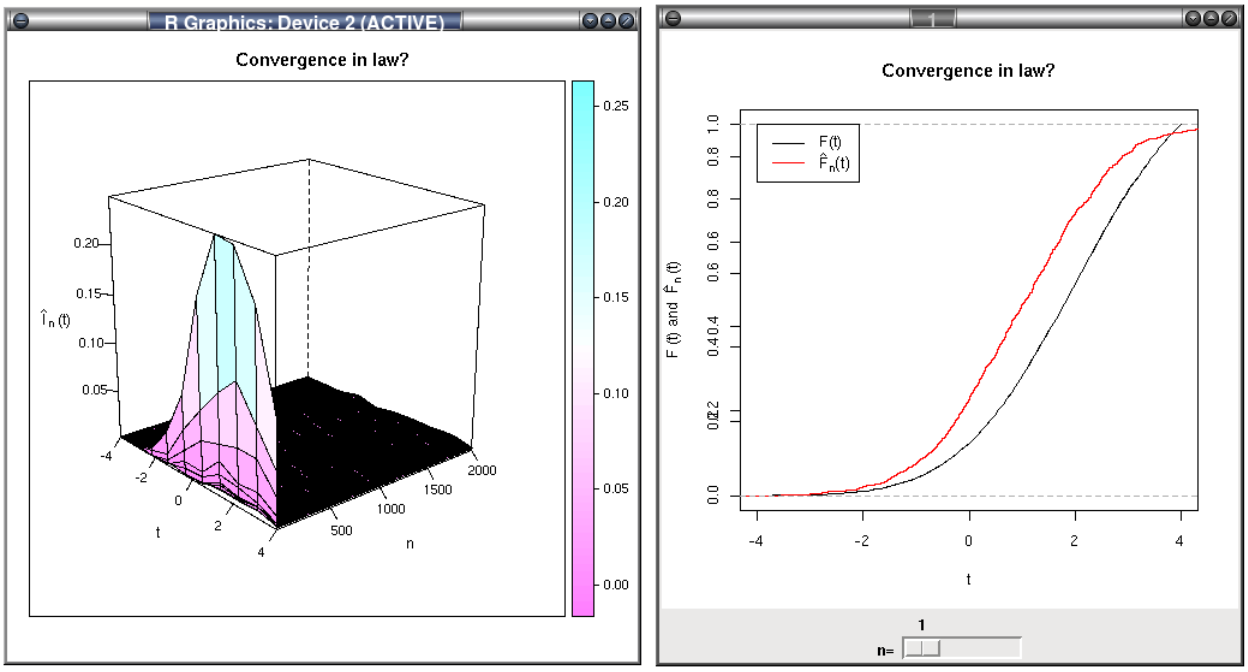

Figure 8: Investigating convergence in law.

$F_n(t)$ is the distribution function of X_n. We define $\hat{F}_n(t)$ to be the empirical distribution function of X_n (based on M realizations of X_n) and $\hat{l}_n(t) = |\hat{F}_n(t) - F(t)|$.

We can move the slider (right part of Figure 8) and see that the red curve comes closer to the black one. Also, on the right you can see the tri-dimensional plot of $|\hat{F}_n(t) - F(t)|$ for $n = 1, \ldots, n_{\max} = 2000$ to see if gets closer to the zero horizontal plane. These plots suggest a convergence in distribution.

Problem 2: Let $X_1, X_2, \ldots$ be *i.i.d.* random variables with a uniform distribution on $[0,1]$. We define $M_n = \max\{X_1, \ldots, X_n\}$.

(a) Show that M_n converges in probability and almost surely to 1.

(b) Show that M_n converges in quadratic mean to 1.

Solution to problem 2:

We first define our random generator of the $(X_i - 1)$'s.

```
genMn <- function(n) {
  res <- cummax(runif(n))-1
  return(res)
}
```

(a) We now call the `check.convergence` function.

```
check.convergence(2000,500,genMn,mode="p")
```

Obviously, all the sample paths are strictly increasing towards 1. Looking at the right panel of Figure 9, we see $\hat{a}_n$ and $\hat{p}_n$ decreasing towards 0. This makes us believe that we are contemplating a convergence almost surely and convergence in probability towards 1.

(b) We now call the `check.convergence` function to investigate the quadratic mean convergence.

```
check.convergence(2000,500,genMn,mode="r",r=2)
```

By definition, $X_n \xrightarrow{r} X$ if and only if $e_{n,r} = E|X_n - X|^r \underset{n\to\infty}{\longrightarrow} 0$. We define, in an obvious fashion, $\hat{e}_{n,r}$ to be a Monte Carlo estimate of $e_{n,r}$, precisely defined in Lafaye de Micheaux and Liquet (2009).

Looking at Figure 10, one can expect M_n to converge in quadratic mean towards 1 since $\hat{e}_{n,r}$ is decreasing towards 0.

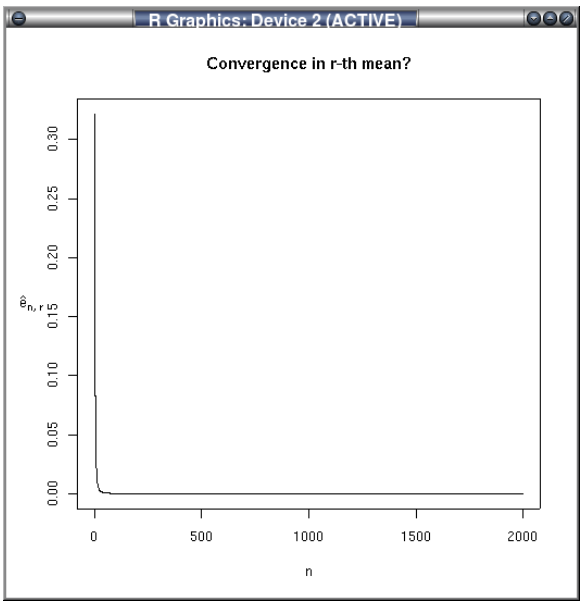

Figure 10: $\hat{p}_n$ and $\hat{a}_n$ going towards 0.

Conclusion

We have described how this package can be used as interactive support for asymptotics courses. A few examples were given to show how to investigate almost sure convergence, convergence in probability, convergence in law, or in r-th mean.

Bibliography

P. Billingsley. *Probability and Measure*. Wiley, New York, 2nd edition, 1986.

G.R. Bryce, R. Gould, W. Notz and R. Peck, R. Curriculum guidelines for bachelor of science degrees in statistical science. *The American Statistician*, 55: 7–13, 2001.

B. Chance and A. Rossman. *Investigating Statistical Concepts, Applications, and Methods*. Duxbury, Belmont, 2006.

K.L. Chung. *A Course in probability theory*. Academic press, New York, 2nd edition, 1974.

P.K. Dunn. Three tools for interactively visualizing some distribution theory concepts. *The American Statistician*, 53:137–139, 1999.

T. Ferguson. *A Course in Large Sample Theory*. Chapman and Hall, 1996.

P. Lafaye de Micheaux and B. Liquet. Understanding convergence concepts: A visual-minded and graphical simulation-based approach. *The American Statistician*, 63(2):173–178, 2009.

E. Lehmann. *Elements of Large-Sample Theory*. Springer, 2nd edition, 2001.

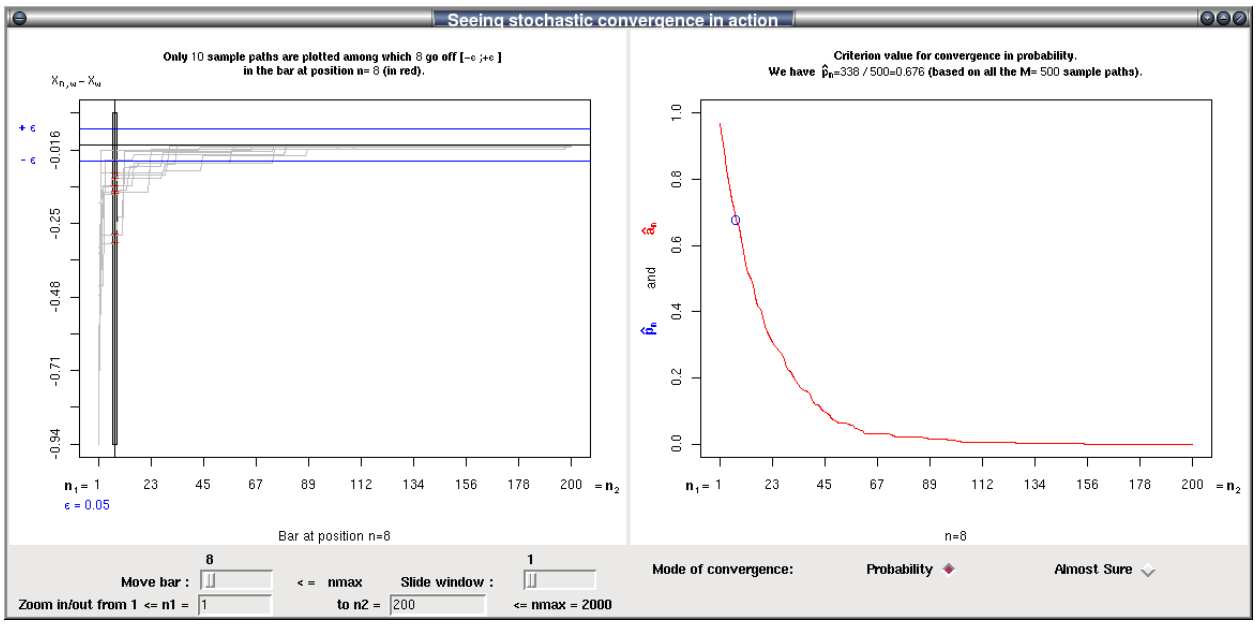

Figure 9: Investigating the convergence in probability and almost surely towards 1 for the sequence of random variables M_n defined in Problem 2. On the right panel, we see the superimposed curves of $\hat{p}_n$ and $\hat{a}_n$ decreasing towards 0.

M.G. Marasinghe, W.Q. Meeker, D. Cook and T. Shin. Using Graphics and Simulation to Teach Statistical Concepts. *The American Statistician*, 50:342–351, 1996.

J.D. Mills. Using computer simulation methods to teach statistics: A review of the literature. *Journal of Statistics Education*, 10(1), 2002. URL http://www.amstat.org/publications/jse/v10n1/mills.html.

R.J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, New York, 2002

J. Sethuraman. Review: Large-sample methods in statistics: An introduction with applications. *Journal of The American Statistical Association*, 90:384, 1995.

Pierre Lafaye de Micheaux
Département de Mathématiques et Statistique
Université de Montréal
Canada
http://www.biostatisticien.eu
lafaye@dms.umontreal.ca

Benoit Liquet
INSERM U897, ISPED
University Bordeaux 2
France
Benoit.Liquet@isped.u-bordeaux2.fr

asympTest: A Simple R Package for Classical Parametric Statistical Tests and Confidence Intervals in Large Samples

by J.-F. Coeurjolly, R. Drouilhet, P. Lafaye de Micheaux and J.-F. Robineau

Abstract: **asympTest** is an R package implementing large sample tests and confidence intervals. One and two sample mean and variance tests (differences and ratios) are considered. The test statistics are all expressed in the same form as the Student t-test, which facilitates their presentation in the classroom. This contribution also fills the gap of a robust (to non-normality) alternative to the chi-square single variance test for large samples, since no such procedure is implemented in standard statistical software.

Introduction

It is sometimes desirable to compare two variances rather than two averages. To cite a few examples (Dean and Illowsky (2009)): college administrators would like two college professors grading exams to have the same variation in their grading; in order for a lid to fit a container, the variation in the lid and the container should be the same; a supermarket might be interested in the variability of check-out times for two checkers.

Now usually, a first course on statistical inference presents mean tests in both Gaussian and asymptotical frameworks (Table 1), but variance tests are often presented only in the Gaussian case (Table 2).

Population law		Test statistic	Law
Gaussian	σ^2 known	$\frac{\overline{Y}_n - \mu_{ref}}{\sigma/\sqrt{n}}$	$\mathcal{N}(0,1)$
	σ^2 unknown	$\frac{\overline{Y}_n - \mu_{ref}}{S_n/\sqrt{n}}$	$t(n-1)$
Unknown ($n > 30$)	σ^2 known	$\frac{\overline{Y}_n - \mu_{ref}}{\sigma/\sqrt{n}}$	$\approx\mathcal{N}(0,1)$ asympt.
	σ^2 unknown	$\frac{\overline{Y}_n - \mu_{ref}}{S_n/\sqrt{n}}$	$\approx\mathcal{N}(0,1)$ asympt.

Table 1: Testing $H_0 : \mu = \mu_{ref}$ for both the Gaussian and large sample cases.

Test statistic	Law
$(n-1)S_n^2/\sigma_{ref}^2$	$\chi^2_{(n-1)}$
S_1^2/S_2^2	$F_{(n_1-1,n_2-1)}$

Table 2: Testing $H_0 : \sigma^2 = \sigma_{ref}^2$ or $H_0 : \sigma_1^2 = \sigma_2^2$ for the Gaussian case (σ^2, σ_1^2, σ_2^2 unknown; σ_{ref}^2 known).

An important point to be noticed is that students are usually told that mean tests are robust to non-normality for large samples as indicated by the asymptotic $\mathcal{N}(0,1)$ distribution in the last two cells of Table 1 (see e.g. Ozgur and Strasser (2004)). They could think that this also occurs for variance tests. Indeed, many practitioners use the classical chi-square single variance test or Fisher's two variances test, even if the Gaussian assumption fails. This could lead to heavy errors, **even for large samples**, as shown in Figure 1. Miller (1997, p. 264) describes this situation as "catastrophic".

To have a better idea of the type I error in the classical single variance test, let us test for example $H_0 : \sigma^2 = 1$ versus $H_1 : \sigma^2 < 1$, by simulating 10000 samples of size 1000 from an $\mathcal{E}(1)$ distribution (i.e. under H_0) and using $\alpha = 5\%$. We obtained a percentage of rejection of the null of 21.53%, thus showing a type I error far greater than α. The percentage for the asymptotic test (described later) is 9.05% which is not too far from α. For a $\mathcal{U}([0,5])$, the classical single variance test leads to a type I error far lesser than α (0.44%). Our test still behaves correctly with a type I error near α (5.39%). This is mainly due to the departure of the kurtosis of the distribution from 3 (for more theoretical details see *e.g.* Section 2.2 of Coeurjolly et al. (2009)).

Note that the problem of the robustness (to departures from normality) of tests for comparing two (or more) variances has been widely treated in the literature, see e.g. Box (1953), Conover et al. (1981), Tiku and Akkaya (2004), Pan (1999) and the references therein. These authors built specific test statistics. Note also that in the one sample (non Gaussian) case, to the best of our knowledge, no statistical tool is available to compare a population variance to a reference value.

Now, it is well-known, see e.g. Casella and Berger (2001, p. 492), that a common method for constructing a large sample test statistic may be based on an estimator that has an asymptotic normal distribution. Suppose we wish to test a hypothesis about a parameter θ, and $\hat{\theta}_n$ is some estimator of θ based on a sample of size n. If we can prove some form of the central limit theorem to show that, as $n \to +\infty$,

$$(\hat{\theta} - \theta)/\hat{\sigma}_{\hat{\theta}} \xrightarrow{d} \mathcal{N}(0,1) \qquad (1)$$

where $\hat{\sigma}_{\hat{\theta}}$ is the usual standard error, which is a convergent (in probability) estimate of $\sigma_{\hat{\theta}} = \sqrt{Var(\hat{\theta}_n)}$, then one has the basis for an approximate test.

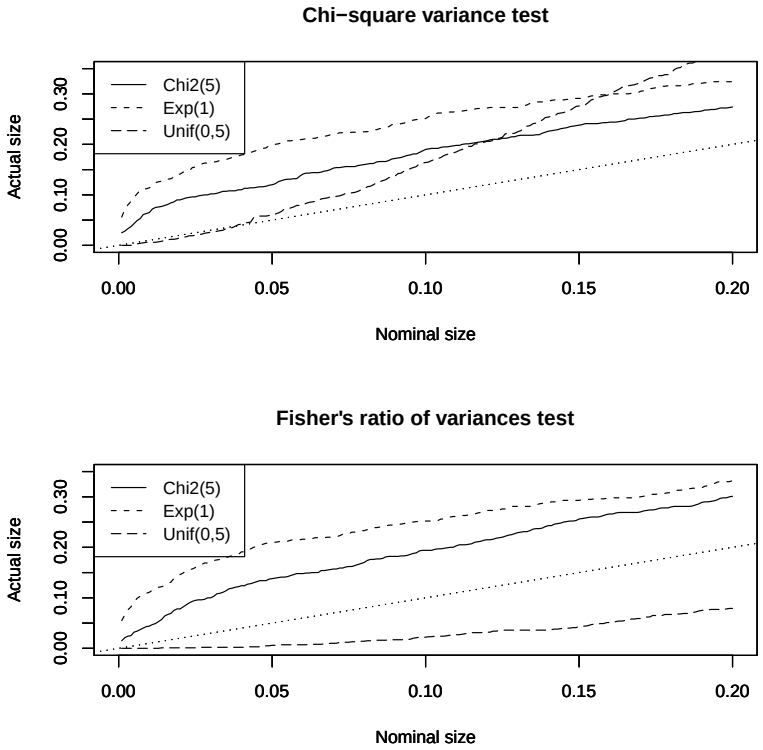

Figure 1: P-value Plots (see Davidson and MacKinnon (1998)) under H_0 of $m = 10000$ replications of test statistics of the chi-square variance test (top) and Fisher's ratio of variances test (bottom) in the large sample Gaussian context. The parameters of the simulation are: $n = n_1 = n_2 = 500$, $Y \overset{d}{=} Y_1 \overset{d}{=} Y_2 \sim \chi^2(5)$ (resp. $\mathcal{E}(1)$, resp. $\mathcal{U}[0,5]$). The dotted lines are 45° lines.

This approach can be used to complete Table 2 for the large sample case, shown in Table 3 for the single variance test only:

Population law	Test statistic	Law
Unknown (n large) with finite 4^{th} moment	$\dfrac{S_n^2 - \sigma_{ref}^2}{\hat{\sigma}_{S_n^2}}$	$\approx \mathcal{N}(0,1)$ asympt.

Table 3: Testing $H_0 : \sigma^2 = \sigma_{ref}^2$ for the large sample case. We let $\hat{\sigma}^2_{S_n^2} = \frac{1}{n(n-1)} \sum_{i=1}^n ((y_i - \overline{Y}_n)^2 - S_n^2)^2$.

The case of a (large sample) test for a difference in scale parameters (possibly weighted by a factor ρ) is also of interest as suggested by the availability of related procedures in R (to compute Ansari-Bradley's and Mood's tests for example). The standard error involved in this test is $\hat{\sigma}_{\hat{\theta}} = \sqrt{\hat{\sigma}^2_{S^2_{n_1}} + \rho^2 \hat{\sigma}^2_{S^2_{n_2}}}$.

The point to be noted here is that this general approach has been extensively used in Coeurjolly et al. (2009) where we end up with a unified approach very similar to the classical t-test from a mathematical point of view. Proofs, which are not very complicated, are provided in the report just cited. The details are not fully expounded here but lead us to propose a more complete, homogeneous teaching framework, with no additional difficulty, to test various parameters such as the mean, the variance, and the difference or ratio of means or variances (for large samples). This approach also allows the direct derivation of asymptotic confidence intervals. Note that Bonnet (2006a) and Bonnet (2006b) use a similar asymptotic approach, with a refinement based on a variance stabilizing transformation, to obtain asymptotic confidence intervals, solely for the single variance and ratio of variances cases. Table 4 gives a summary of the various parameters we can test and the R functions we have implemented to compute the standard error $\hat{\sigma}_{\hat{\theta}}$ of $\hat{\theta}$:

θ	Dataset(s)	$\hat{\sigma}_{\hat{\theta}}$ in R
μ	`y`	`seMean(y)`
σ^2	`y`	`seVar(y)`
$d_\mu = \mu_1 - \rho\mu_2$	`y1, y2`	`seDMean(y1,y2,rho)`
$d_{\sigma^2} = \sigma_1^2 - \rho\sigma_2^2$	`y1, y2`	`seDVar(y1,y2,rho)`
$r_\mu = \mu_1/\mu_2$	`y1, y2`	`seRMean(y1,y2)`
$r_{\sigma^2} = \sigma_1^2/\sigma_2^2$	`y1, y2`	`seRVar(y1,y2)`

Table 4: Various parameters we can test and available R functions to compute standard error $\hat{\sigma}_{\hat{\theta}}$.

These functions can be used in conjunction with (1) to obtain p-values for various tests. For a simple example, if you want to use a sample contained in

	$\mathcal{E}(1)$		$\chi^2(5)$		$\mathcal{U}([0,5])$	
n	χ^2	asymp.	χ^2	asymp.	χ^2	asymp.
30	0.2168	0.2733	0.1278	0.2218	0.0086	0.0801
100	0.2194	0.1765	0.1307	0.1442	0.0061	0.0589
500	0.2157	0.1102	0.1367	0.0928	0.0051	0.0543
1000	0.2153	0.0905	0.1323	0.0787	0.0040	0.0539

Table 5: Type I error in terms of n for the test $H_1 : \sigma^2 < \sigma^2_{ref}$ with $\sigma^2_{ref} = 1$ ($\mathcal{E}(1)$), 10 ($\chi^2(5)$), 25/12 ($\mathcal{U}([0,5])$) based on $m = 10000$ replications.

the vector `y` to test $H_0 : \sigma^2 = 1$, you can use

```
2*pnorm(-abs((var(y)-1)/seVar(y)))
```

This contribution also solves the problem of providing an implemented "robust" (to departure of the i.i.d. large sample distribution from normality) alternative to the chi-square single variance test for large samples. Indeed, we did not find any such procedure in standard statistical software and so it is highly likely that practitioners would incorrectly use a chi-square test on a single variance. It also provides a very simple alternative to the (ratio of variances) Fisher test in large samples. Some other "robust" alternative procedures to the Fisher test in the case of non Gaussian (not necessary large) samples are implemented in R: the Bartlett test (`bartlett.test`), the Fligner test (`fligner.test`) and the Levene test (`levene.test` available in the `lawstat` package). R also provides, through `ansari.test` and `mood.test` functions, Ansari-Bradley's and Mood's two-sample rank-based tests for a difference in scale parameters. The purpose of this paper is not to compare our tests to their competitors in terms of power. We nevertheless conduct two short simulation studies (limited to the probability of Type I error): first for the problem of testing a variance (Table 5), comparing the classical χ^2 single variance test to our procedure, and second for the problem of comparing (the differences d_{σ^2} of) two variances (Tables 6, 7 and 8), comparing the classical Fisher test to our procedure, as well as Ansari-Bradley's test and Mood's test. These simulations were based on the three distributions used earlier in Figure 1. The simulations show that the level α is quite correct (when n increases) for our procedure in the case of testing a single variance and for all three alternative tests (ours, Ansari-Bradley's and Mood's tests) for testing two variances.

	$\mathcal{E}(1)$			
n	$\mathcal{F}$	asympTest	Ansari	Mood
30	0.2827	0.0675	0.0478	0.0497
100	0.3083	0.0500	0.0480	0.0484
500	0.3269	0.0454	0.0484	0.0470
1000	0.3260	0.0526	0.0501	0.0515

Table 6: Type I error for the test $H_1 : \sigma^2_1 \neq \sigma^2_2$ in terms of n for $m = 10000$ replications of the distribution $\mathcal{E}(1)$.

	$\chi^2(5)$			
n	$\mathcal{F}$	asympTest	Ansari	Mood
30	0.1605	0.0676	0.0477	0.0472
100	0.1797	0.0537	0.0516	0.0494
500	0.1911	0.0525	0.0505	0.0498
1000	0.1907	0.0526	0.0503	0.0511

Table 7: Type I error for the test $H_1 : \sigma^2_1 \neq \sigma^2_2$ in terms of n for $m = 10000$ replications of the distribution $\chi^2(5)$.

	$\mathcal{U}([0,5])$			
n	$\mathcal{F}$	asympTest	Ansari	Mood
30	0.0029	0.0652	0.0490	0.0494
100	0.0021	0.0527	0.0490	0.0475
500	0.0024	0.0520	0.0511	0.0511
1000	0.0022	0.0539	0.0528	0.0538

Table 8: Type I error for the test $H_1 : \sigma^2_1 \neq \sigma^2_2$ in terms of n for $m = 10000$ replications of the distribution $\mathcal{U}([0,5])$.

Using asympTest

The R package **asympTest** consists of a main function `asymp.test` and six auxiliary ones designed to compute standard errors of estimates of different parameters, see Table 4. The auxiliary functions will not be the most useful ones for the user, except if he/she wants to compute the confidence interval himself/herself. The function `asymp.test` has been written in the same spirit as the standard R functions `t.test` or `var.test`. The arguments of `asymp.test` and the resulting outputs are also inspired from these functions. In particular, the function `asympt.test` returns an object of class "htest" (which is the general class of test objects in R).

This `asymp.test` function has several arguments, similar to those of the `t.test` function, whose description can be obtained using the command `?asymp.test`.

In order to illustrate this function, let us consider the Digitalis Investigation Group NHLBI Teaching data set (`https://biolincc.nhlbi.nih.gov/teaching/`) which was made available by the NHLBI. Note that statistical processes such as permutations within treatment groups were used to

completely anonymize the data; therefore, inferences derived from the teaching dataset may not be valid.

The DIG Trial was a randomized, double-blind, multicenter trial with more than 300 centers in the United States and Canada participating. The purpose of the trial was to examine the safety and efficacy of Digoxin in treating patients with congestive heart failure in sinus rhythm.

Diastolic BP (`DIABP`, mmHg) is a known risk factor of cardiovascular diseases. In this case, it is desirable to compare the variability of this quantity for placebo (`TRTMT=0`) and treatment (`TRTMT=1`) groups, respectively.

Reading of the data

```
> require(asympTest)
>
> data(DIGdata)
> attach(DIGdata)
> x <- na.omit(DIABP[TRTMT==0])
> y <- na.omit(DIABP[TRTMT==1])
> c(length(x),length(y))
[1] 3400 3395
```

Comparing the two variances

Shapiro-Wilk normality test performed by the function `shapiro.test()` indicates that the two samples seem to be far from the Gaussian distribution. Thus, this should prevent us from using the following Fisher test.

```
> var.test(DIABP ~ TRTMT, data = DIGdata,
+    na.action = na.omit)

        F test to compare two variances

data:  x and y
F = 0.9295, num df = 3399, denom df = 3394
p-value = 0.03328
alternative hypothesis:
   true ratio of variances is not equal to 1
95 percent confidence interval:
 0.8690651 0.9942238
sample estimates:
ratio of variances
          0.929541
```

Instead, let us use our package.

```
> asymp.test(DIABP ~ TRTMT, data = DIGdata,
+    na.action = na.omit, parameter = "dVar")

Two-sample asymptotic diff. of variances test

data:  DIABP by TRTMT
statistic = -1.5272, p-value = 0.1267
alternative hypothesis:
  true diff. of variances is not equal to 0
95 percent confidence interval:
 -21.160491   2.626127
sample estimates:
difference of variances
             -9.267182
```

We can see that `var.test`, not to be used due to the unlikely normality of the data, significantly shows a difference in variances (at a 5% level). We don't obtain the same conclusion with our test.

We can also place ourselves in a fictitious case by generating a sample `x` from a $\mathcal{U}(0;\sqrt{12})$ (i.e. with a true population variance $\sigma^2 = 1$). We then apply both our test and the classical chi-square test to show $H_1 : \sigma^2 > \sigma^2_{ref} = 0.97$.

```
> n <- 1000
> x <- runif(n, max = sqrt(12))
> asymp.test(x, par = "var", alt = "gr",
+    ref = 0.97)

        One-sample asymptotic variance test

data:  x
statistic = 1.753, p-value = 0.0398
alternative hypothesis:
         true variance is greater than 0.97
95 percent confidence interval:
 0.9731491       Inf
sample estimates:
variance
1.021055
> chisq.stat <- (n-1)*var(x)/0.97
> pchisq(chisq.stat, n-1, lower.tail = F)
[1] 0.1207650
```

For the above generated sample `x`, we respectively found the following p-values: 0.0398 and 0.120. In this case, we can thus see that our proposition correctly accepts H_1 (at the 5% level) but not the chi-square single variance test.

Conclusion

This paper has introduced a new package called **asympTest**. This is a contribution to the many R procedures available. It is interesting firstly in the fact that it provides a unified teaching framework to present classical parametric tests (based on the Central Limit Theorem). These tests are made readily available in R through an easy to use function called `asymp.test`. This function resembles `t.test` or `var.test`, so students will not be confused. Secondly, it also makes available in R a robust (to non-normality) alternative to the classical chi-square single variance test. In the future, we also plan to provide tools similar to the `power.t.test` function in the context of large samples.

Bibliography

D. G. Bonnet. Approximate confidence interval for standard deviation of nonnormal distributions. *Computational Statistics & Data Analysis*, 50:775–782, 2006.

D. G. Bonnet. Robust confidence interval for a ratio of standard deviations. *Applied Psychological Measurement*, 30:(5) 432–439, 2006.

G. E. P. Box. Non-normality and tests on variances. *Biometrika*, 40:(3/4) 318–335, 1953.

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, Belmont, California, 2nd edition, 2001.

J.-F. Coeurjolly, R. Drouilhet, P. Lafaye de Micheaux and J.-F. Robineau. asympTest: an R package for performing parametric statistical tests and confidence intervals based on the central limit theorem. Technical report, URL http://www.citebase.org/abstract?id=oai:arXiv.org:0902.0506.

W. J. Conover, M. E. Johnson and M. M. Johnson. A comparative study of tests for homogeneity of variances with applications to the outer continental shelf bidding data. *Technometrics*, 23:(4) 351–361, 1981.

R. Davidson and J. G. MacKinnon. Graphical methods for investigating the size and power of hypothesis tests. *The Manchester School*, 66:(1) 1–26, 1998.

S. Dean and B. Illowsky. F Distribution and ANOVA: Test of Two Variances. *Connexions*, February 5, 2009. URL http://cnx.org/content/m17075/1.6/

T. Ferguson. A course in large sample theory. *Chapman and Hall*, 1996.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:(Part II) 179–188, 1935.

R. G. Miller. *Beyond ANOVA, Basics of Applied Statistics*. Texts in Statistical Science Series. Chapman & Hall/CRC, 1997.

C. Ozgur and S. E. Strasser. A study of the statistical inference criteria: Can we agree on when to use z versus t? *Decision Sciences Journal of Innovative Education*, 2:(2) 177–192, 2004.

G. Pan. On a Levene type test for equality of two variances. *Journal of Statistical Computation and Simulation*, 63:59–71, 1999.

M. L. Tiku and A. Akkaya. *Robust Estimation and Hypothesis Testing*. New Age International (P) Ltd, New Delhi, 2004.

Jean-François CoeurJolly
L. Jean Kuntzmann, Grenoble University, France
http://cqls.upmf-grenoble.fr
Jean-Francois.Coeurjolly@upmf-grenoble.fr

Rémy Drouilhet
L. Jean Kuntzmann
Grenoble University
France
http://cqls.upmf-grenoble.fr
Remy.Drouilhet@upmf-grenoble.fr

Pierre Lafaye de Micheaux
Département de Mathématiques et Statistique
Université de Montréal
Canada
http://www.biostatisticien.eu
lafaye@dms.umontreal.ca

Jean-François Robineau
CQLS
http://www.cqls.org
robineau@cqls.fr

copas: An R package for Fitting the Copas Selection Model

by J. Carpenter, G. Rücker and G. Schwarzer

Abstract: This article describes the R package **copas** which is an add-on package to the R package **meta**. The R package **copas** can be used to fit the Copas selection model to adjust for bias in meta-analysis. A clinical example is used to illustrate fitting and interpreting the Copas selection model.

Introduction

Systematic reviews play a key role in the evaluation of new interventions. While combining evidence from a number of studies should reduce both bias and uncertainty, this is sometimes not the case because published studies represent a biased selection of the evidence. This can happen for a number of reasons, for example authors may be more likely to submit trials with 'significant' results for publication or journals may be more likely to publish smaller trials if they have 'significant' results. Empirical studies have established evidence for these kinds of biases and others (Rothstein et al., 2005; Nieminen et al., 2007).

In consequence, considerable effort has been directed at the problem of developing reliable tests for selection biases (Harbord et al., 2006; Rücker et al., 2008), and in a second step, correcting estimates for publication bias (Rothstein et al., 2005). One of the most promising methods to date has been the so-called Copas selection model (Copas, 1999; Copas and Shi, 2000, 2001), which is derived from the Heckman 2-stage regression model (see Little and Rubin (2002)).

Comprehensive evaluations suggest that this approach (i) can provide a useful summary in around 80% of meta-analyses (Carpenter et al., 2009) and (ii) is preferable to the trim-and-fill method to adjust for bias in meta-analysis (Schwarzer et al., 2009). This article describes the R package **copas** for fitting and interpreting the Copas selection model.

Copas selection model

We first briefly describe the model. The Copas selection model has two components: (i) a model for the outcome, measured on a chosen scale, e.g. the log odds ratio, log risk ratio or the arcsine difference (Rücker et al., 2009), and (ii) a 'selection' model giving the probability that study i is observed/published. A correlation parameter ρ between these two components models the extent of selection/publication bias; the stronger the correlation, the greater the chance that only more extreme outcomes are observed/published.

In more detail, let (ϵ_i, δ_i) follow a bivariate normal distribution with mean 0 and covariance matrix

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

Denote the underlying population intervention effect by θ, and between study heterogeneity variance τ^2. For each study i in a meta-analysis, let $\hat{\theta}_i$ denote the estimate of θ and s_i^2 the estimated variance of $\hat{\theta}_i$, whose true, unobserved, variance is σ_i^2. We model the observed outcome of study i using the usual random effects model, i.e. as

$$\hat{\theta}_i = \theta + \sqrt{(\tau^2 + \sigma_i^2)}\epsilon_i \tag{1}$$

and say study i is observed/published if $Z_i > 0$, where

$$Z_i = \gamma_0 + \gamma_1 / s_i + \delta_i \tag{2}$$

with fixed γ_0 and γ_1. From (2), the marginal probability that study i is observed is

$$\Pr(Z_i > 0) = \Pr(\delta_i > -\gamma_0 - \gamma_1 / s_i) \tag{3}$$

$$= \Phi(\gamma_0 + \gamma_1 / s_i), \tag{4}$$

where $\Phi(.)$ is the cumulative density function of the standard normal. Thus $\Phi(\gamma_0)$ can be interpreted as the marginal probability of publishing a study with infinite standard error, and γ_1 is associated with the change in publication probability with increasing precision. Note that the appearance of s_i in (2) means that the probability of publication reflects the sampling variance of study i.

Copas (1999) and Copas and Shi (2000) use standard properties of the normal distribution to show that the probability of observing study i is

$$\Phi\left\{ \frac{\gamma_0 + \gamma_1 / s_i + \rho\sigma_i(\hat{\theta}_i - \theta)/(\sigma_i^2 + \tau^2)}{\sqrt{1 - \rho^2\sigma_i^2/(\sigma_i^2 + \tau^2)}} \right\}. \tag{5}$$

Thus if $\rho = 0$, (1) and (2) are unrelated and a meta-analysis of observed studies will give an approximately unbiased estimate of θ. Conversely, if large θ means a strong treatment effect and $\rho > 0$, then the probability of observing study i is increased the larger $\hat{\theta}_i$. In this situation, a meta-analysis of observed studies will give a biased estimate of θ.

Fitting the Copas selection model

We have developed the R package **copas** to provide a comprehensive set of R functions for fitting the

```
> ## Read in data
> data(Crowther2003)
> ## Do meta-analysis using function metabin() from R package meta
> m.crowther <- metabin(event.e, total.e, event.c, total.c,
+                       data=Crowther2003, sm="OR", studlab=study)
Warning message:
In metabin(event.e, total.e, event.c, total.c, data = Crowther2003,  :
  Increment 0.5 added to each cell in 2x2 tables with zero cell frequencies
> ## Do test for funnel plot asymmetry (Harbord et al.)
> ## using function metabias() from R package meta
> metabias(m.crowther, meth="score")

Linear regression test of funnel plot asymmetry (efficient score)

data:  m.crowther
t = -3.4551, df = 7, p-value = 0.01062
alternative hypothesis: asymmetry in funnel plot
sample estimates:
      bias    se.bias      slope
-2.6149672  0.7568465  0.2927025

> ## Do Copas analysis
> cop1 <- copas(m.crowther)
> ## Plot Copas analysis
> plot(cop1)
> ## Redo Copas analysis as P-value of residual selection bias
> ## is still significant
> cop2 <- copas(m.crowther,
+               gamma0.range=c(-0.55, 2),
+               gamma1.range=cop1$gamma1.range)
Warning messages:
1: In sqrt(1 - rho.tilde^2) : NaNs produced
2: In sqrt((tau^2 + sigma^2) * (1 - rho.tilde^2)) : NaNs produced
3: In sqrt((1 - rho.tilde^2)) : NaNs produced
> ## Plot Copas analysis
> plot(cop2)
> ## Print summary of Copas analysis
> summary(cop2)
Summary of Copas selection model analysis:

             publprob     OR            95%-CI pval.treat  pval.rsb N.unpubl
                 1.00 0.4967  [0.3247; 0.7599]    0.0013    0.006          0
                 0.82 0.5483  [0.3494; 0.8605]    0.009     0.007          1
                 0.67 0.6063  [0.3938; 0.9335]    0.023     0.0115         2
                 0.55 0.6702  [0.4601; 0.9761]    0.037     0.0205         4
                 0.45 0.7402  [0.5376; 1.0193]    0.0653    0.046          6
                 0.37 0.8337  [0.6055; 1.1480]    0.2652    0.2461         9

    Copas model (adj) 0.8337  [0.6055; 1.1480]    0.2652    0.2461         9
 Random effects model 0.4880  [0.3234; 0.7363]    0.0006

 Legend:
 publprob   - Probability of publishing the study with the largest standard error
 pval.treat - P-value for hypothesis that the treatment effect is equal in both groups
 pval.rsb   - P-value for hypothesis that no further selection remains unexplained
 N.unpubl   - Approximate number of studies the model suggests remain unpublished
```

Figure 1: Example of Copas analysis of phenobarbital versus control for reducing neonatal periventricular haemorrhage (Crowther and Henderson-Smart, 2003); output of function `summary.copas()`.

Copas selection model, then printing and displaying the results. The R package **copas** is an add-on package to the R package **meta** (Schwarzer, 2007). To illustrate the use of the R package **copas**, consider a meta-analysis of 9 studies comparing prophylactic maternal phenobarbital with control for periventricular haemorrhage in preterm infants (Crowther and Henderson-Smart, 2003).

Figure 1 illustrates the analysis of these data, which have a binary response (presence or absence of haemorrhage). After reading the data in, we perform a meta-analysis on the odds-ratio scale using function `metabin()` from the R package **meta**. Note the warning that a continuity correction has been used, since one study has a zero cell. By default, 0.5 is added only to cell counts of two-by-two tables with zero cell frequencies. Two other stategies to adjust for zero cell frequencies are implemented in the function `metabin()` (Schwarzer, 2007): (i) add 0.5 to all two-by-two tables in the case of zero cell counts in one or more studies (`allincr = TRUE`), (ii) add 0.5 to all two-by-two tables irrespective of zero cell counts (`addincr = TRUE`).

The random effects model suggests a significant intervention benefit (p = 0.0006). Nevertheless, a funnel plot of the data (top left panel Figure 2) suggests effects may be systematically larger in smaller studies. Investigating this with a statistical test (function `metabias()`, Figure 1) supports this suspicion with $p = 0.011$.

We therefore perform a Copas analysis, using the function `copas()`, as shown in Figure 1. This fits the Copas selection model repeatedly, by maximising the likelihood (Carpenter et al., 2009) — subject to the constraints that $-1 < \rho < 1$ and $\tau^2 \geq 0$ — over a grid of (γ_0, γ_1) values using the existing R function `optim()` (L-BFGS-B method). We use transformations of ρ and τ^2 in the log-likelihood to reduce numerical instability, which is a known issue with this model (Carpenter et al., 2009). We redo the Copas analysis extending the range of values for γ_0 as the *P*-value of residual selection bias is still significant using the default settings. Three warning messages are printed concerning a parameter `rho.tilde` which is used internally in likelihood estimation (Copas and Shi, 2000, p. 250). Typically, these warnings can be safely ignored. The function `plot.copas()` can be used as a diagnostic tool (see Discussion).

An object of class `copas` is created which we can `print`, `summarise` and `plot`. Figure 2 shows the result of function `plot.copas()`, which by default displays four panels. The top left plot is the usual funnel plot, which plots the study specific effect sizes (here log odds ratios) against their standard error. The vertical broken line is the usual fixed effect estimate of the treatment effect, while the vertical grey line is the usual random effects estimate of the treatment effect. Diagonal broken lines show $\pm$ 2 standard errors about the fixed effect. If there is no heterogeneity or selection bias, we would expect about 95% of studies to lie in this 'funnel'. In this example, there is a suggestion that smaller studies tend to show a stronger effect.

Given the range of study standard errors in the meta-analysis, the function `copas()` chooses a range of γ_0 and γ_1 values for the selection model (2). These are chosen to represent varying selection strength. Specifically, the probability of publishing the study with the largest SE (often the smallest study) ranges from around 0.3 to 1. In this example, the initial analysis (Figure 1, with the default ranges for (γ_0, γ_1)) has the lower bound for γ_0 at -0.45, which just stops short of the region where the degree of selection is sufficient to explain the asymmetry in the funnel plot. We thus repeat the analysis with a slightly larger range for γ_0, and it is this second analysis that is shown in Figure 2.

The function `copas()` fits the Copas selection model over a grid of (by default 400) points. The top right panel produced by `plot.copas` shows a contour plot of the resulting treatment estimates $(\hat{\theta})$ over the range of (γ_0, γ_1) values. Contours are labeled with values of $\hat{\theta}$, (which can be specified by the user) in this case $-0.6, -0.5, -0.4, \ldots$. The contour plot suggests that as selection increases (i.e. as we move away from the top right) the treatment estimate declines, but the contours are locally parallel. As described in more detail in the Appendix of Carpenter et al. (2009), in such cases the contour plot can be summarised by looking at how the treatment estimate varies as selection increases along a line orthogonal to the contours. Using an algorithm we developed (Carpenter et al., 2009), this orthogonal line is estimated and superimposed on the contour plot. The places where it intersects with the contours are marked with an 'o'.

The lower two panels of Figure 2 use this information to present an accessible summary of the Copas selection model analysis. First, at each of the line/contour intersections marked with an 'o' in the contour plot, the program calculates the probability of publishing the trial with the largest SE. Then, in the lower left panel this is plotted against the corresponding treatment estimate ($\pm$ its 95% confidence interval). In our example this shows that with little or no selection, the treatment estimate is close to that obtained by the usual random effects meta-analysis.[1] As selection increases (so studies with large SE's but smaller effect sizes are less likely to make it into the meta-analysis) so the treatment estimate moves towards the null value and the significance (indicated by the degree that the 95% confidence interval overlaps 0) decreases.

Finally, the bottom right panel attempts to answer

[1] Exact agreement is not to be expected, as the usual random effects analysis uses a method of moments estimate of heterogeneity, whereas the Copas selection model uses a maximum likelihood estimate.

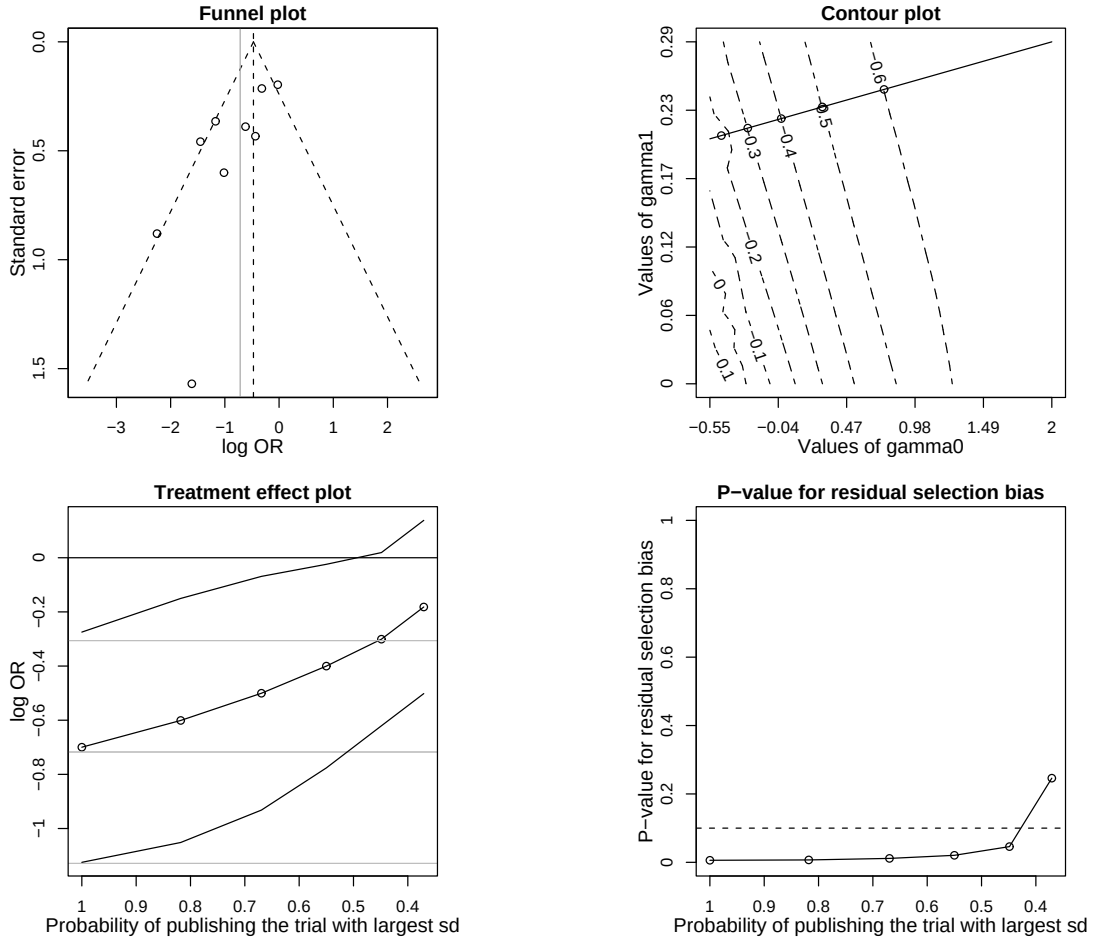

Figure 2: Plot of the results of a `copas` analysis of phenobarbital versus control for reducing neonatal periventricular haemorrhage (Crowther and Henderson-Smart, 2003).

the question of which degree of selection is most plausible under the model; i.e. which treatment estimates should we pay most attention to. It does this as follows. For each degree of selection (summarised by the probability of publishing the trial with the largest SE) the program calculates whether the remaining asymmetry in the funnel plot is more than might be expected by chance, and calculates a p-value to test this hypothesis. These are then plotted against the corresponding degree of selection. Degrees of selection corresponding to p-values above 0.1 (i.e. at this degree of selection, no evidence of residual selection bias in the data) are more plausible under the model; corresponding treatment estimates in the bottom left panel are more plausible.

In this example, with no selection bias (left hand end of bottom left panel) the p-value for residual selection bias in the data is < 0.1, suggesting the meta-analysis should not be interpreted at face value. Rather, it is only when the probability of publishing the trial with the largest SD is as low as 0.4 that the asymmetry seen in the funnel plot is explained. In this case, the bottom left panel indicates the treatment effect is no longer significant at the 5% level. The function `summary.copas()` (Figure 1) complements this with a numerical summary.

The Copas selection model analysis therefore suggests that after accounting for selection bias and/or other small study effects, there is no evidence of a benefit of the intervention. This agrees with the authors of the original study, who comment that the two largest trials — which are of higher quality — show no benefit, and conclude "evidence does not support phenobarbital treatment to women giving birth before 34 weeks to decrease the risk of bleeding into the babies' brains".

Arguments of function `copas()`

Although the majority of studies can be analysed automatically using the Copas selection model this is not always true. Some analyses will require fine tuning. The following options are available:

- `gamma0.range`, `gamma1.range`:
  These allow users to control the grid of (γ_0, γ_1) values in the selection part of the Copas selection model (equation 2) which the program ranges over to produce the contour plot (top right panel Figure 2).

- `ngrid`:
  This parameter controls how fine the grid of (γ_0, γ_1) is. The `copas` function fits the Copas selection model over a grid of `ngrid`×`ngrid` values.

- `levels`:
  Fitting the Copas model over the grid specified by the arguments `gamma0.range`, `gamma1.range` and `ngrid` results in a treatment estimate at every point in the grid. These are then displayed on a contour plot where contours of treatment effect (z-axis) are shown by gamma0 (x-axis) and gamma1 (y-axis). This argument is a numeric vector which specifies the treatment effects for which contour lines will be drawn.

- `left`:
  A logical indicating whether the cause of any selection bias is due to missing studies on the left or right of the funnel plot: left hand side if `left = TRUE`, right hand side if `left = FALSE`. This information is needed in order to be sure the test for presence of residual selection bias is calculated correctly.

- `rho.bound`:
  A number giving the upper bound for the correlation parameter ρ in the Copas selection model. The default is 0.9999. At this value, warnings are sometimes triggered by the program attempting to take the square root of numbers that are just negative; in all analyses we have carried out these can safely be ignored. Alternatively, repeat the analysis with a slightly smaller bound. Values less than 0.95 are likely to cause irregularites in regions of the contour plot where there is a high degree of selection.

- `silent`:
  A logical indicating whether information on progress in fitting the Copas selection model should be printed: `silent = TRUE` specifies not to print information (the default).

- `warn`:
  A number setting the handling of warning messages. It is not uncommon for numerical problems to be encountered during estimation over the grid of (γ_0, γ_1) values. Usually this does not indicate a serious problem. This option specifies what to do with warning messages — `warn = -1`: ignore all warnings; `warn = 0` (the default): store warnings till the function finishes; if there are less than 10, print them, otherwise print a message saying warning messages were generated; `warn = 1`: print warnings as they occur; `warn = 2`: stop the function when the first warning is generated.

All the information used to generate the plots is available as attributes of the object created by the `copas` function. Thus tailor-made versions of the panels in Figure 2 can be created by users without any further calculation.

Finally, the `summary` function and the `plot` function allow the user to specify the confidence level.

Discussion

The extensive literature on selection bias in meta-analysis (see Rothstein et al. (2005) and references therein) reflects the importance to the community of systematic reviewers of detecting, and where possible adjusting for, selection bias. The Copas selection model is a key tool for doing this (Carpenter et al., 2009) (alongside other methods such as the trim-and-fill method included in the **meta** package (Schwarzer, 2007)).

An empirical evaluation of 157 meta-analyses with 4 to 66 studies showed that our implementation of the Copas selection model provided a useful summary in about 80% of meta-analyses (Carpenter et al., 2009). In the remaining meta-analyses (i) the contour plot did not show roughly parallel contour lines, (ii) the 95% confidence intervals in the treatment effect plot did not vary smoothly, or (iii) P-values in the P-value plot for residual selection bias were erratic. A contour plot without roughly parallel contour lines did appear in situations with an apparently symmetric funnel plot, i.e. when there was no indication of selection bias. This is not a weakness of the model or the software but a consequence of the flat likelihood and the treatment effect being invariant in this situation. Irregularities in the treatment effect plot and P-value plot are typically due to estimation problems. In general, problems in the estimation process can be judged by looking at the output from function `plot.copas()` which should be used routinely as a diagnostic tool.

We have attempted to make the help files for the **copas** package accessible to systematic reviewers, who in most cases are likely to be new users of R. With this package, we therefore believe that R has a powerful toolkit for systematic reviewers.

Acknowledgments

This research was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) under grant FOR 534 Schw 821/2-2

Bibliography

J. R. Carpenter, G. Schwarzer, G. Rücker, and R. Künstler. Empirical evaluation showed that the Copas selection model provided a useful summary in 80% of meta-analyses. *Journal of Clinical Epidemiology*, 62:624–631, 2009.

J. Copas. What works?: Selectivity models and meta-analysis. *Journal of the Royal Statistical Society, Series A*, 162:95–109, 1999.

J. Copas and J. Q. Shi. Meta-analysis, funnel plots and sensitivity analysis. *Biostatistics*, 1:247–262, 2000.

J. B. Copas and J. Q. Shi. A sensitivity analysis for publication bias in systematic reviews. *Statistical Methods in Medical Research*, 10:251–265, 2001.

C. A. Crowther and D. J. Henderson-Smart. Phenobarbital prior to preterm birth for preventing neonatal periventricular haemorrhage. *Cochrane Database of Systematic Reviews*, 3, 2003. Art. No.: CD000164, doi: 10.1002/14651858.CD000164.

R. M. Harbord, M. Egger, and J. A. Sterne. A modified test for small-study effects in meta-analyses of controlled trials with binary endpoints. *Statistics in Medicine*, 25(20):3443–3457, 2006.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Chichester, second edition, 2002.

P. Nieminen, G. Rücker, J. Miettunen, J. R. Carpenter, and M. Schumacher. Statistically significant papers in psychiatry were cited more often than others. *Journal of Clinical Epidemiology*, 60(9):939–946, 2007.

H. R. Rothstein, A. J. Sutton, and M. Borenstein. *Publication Bias in Meta Analysis: Prevention, Assessment and Adjustments*. Wiley, Chichester, 2005.

G. Rücker, G. Schwarzer, J. Carpenter, and I. Olkin. Why add anything to nothing? The arcsine difference as a measure of treatment effect in meta-analysis with zero cells. *Statistics in Medicine*, 28 (5):721–738, 2009.

G. Rücker, G. Schwarzer, and J. R. Carpenter. Arcsine test for publication bias in meta-analyses with binary outcomes. *Statistics in Medicine*, 27(5):746–763, 2008.

G. Schwarzer. meta: An R package for meta-analysis. *R News*, 7(3):40–45, 2007. URL http://cran.r-project.org/doc/Rnews/Rnews_2007-3.pdf.

G. Schwarzer, J. R. Carpenter, and G. Rücker. Empirical evaluation suggests Copas selection model preferable to Trim-and-Fill for selection bias in meta-analysis. *Journal of Clinical Epidemiology*, in press, doi: 10.1016/j.jclinepi.2009.05.008.

James Carpenter
Institute of Medical Biometry and Medical Informatics
University Medical Center Freiburg, Germany
jrc@imbi.uni-freiburg.de

Gerta Rücker
Institute of Medical Biometry and Medical Informatics
University Medical Center Freiburg, Germany
ruecker@imbi.uni-freiburg.de

Guido Schwarzer
Institute of Medical Biometry and Medical Informatics
University Medical Center Freiburg, Germany
sc@imbi.uni-freiburg.de

Transitioning to R: Replicating SAS, Stata, and SUDAAN Analysis Techniques in Health Policy Data

by Anthony Damico

Abstract: Statistical, data manipulation, and presentation tools make R an ideal integrated package for research in the fields of health policy and healthcare management and evaluation. However, the technical documentation accompanying most data sets used by researchers in these fields does not include syntax examples for analysts to make the transition from another statistical package to R. This paper describes the steps required to import health policy data into R, to prepare that data for analysis using the two most common complex survey variance calculation techniques, and to produce the principal set of statistical estimates sought by health policy researchers. Using data from the Medical Expenditure Panel Survey Household Component (MEPS-HC), this paper outlines complex survey data analysis techniques in R, with side-by-side comparisons to the SAS, Stata, and SUDAAN statistical software packages.

Introduction

Health-care researchers use survey data to evaluate the state of the market and to inform the creation and implementation of new policies. America's continuing discussion about the health-care system highlights the unique role of population-generalizable data as a gauge for estimating how proposed legislation will affect specific demographic groups and the $2.2 trillion health-care industry.[1] Despite powerful statistical analysis, data manipulation, and presentation capabilities, R has not become widely adopted as a statistical package in the fields of health policy or health-care management and evaluation. User guidelines and other technical documents for government-funded and publicly available data sets rarely provide appropriate code examples to R users. The objective of this paper is to describe the steps required to import health policy data into R, to prepare that data for analysis using the two most common complex survey variance calculation techniques, and to produce the principal set of statistical estimates sought by health policy researchers.

This article compares means and standard errors produced by technical documentation-recommended code examples for selected proprietary statistical packages with those of the **survey** package available in R, using data from the Medical Expenditure Panel Survey–Household Component (MEPS-HC).[2]

Medical Expenditure Panel Survey–Household Component

- Nationally representative (non-institutionalized) sample of health services utilization in the United States
- Administered by the Agency for Healthcare Research and Quality (AHRQ)
- Panel survey design allows tracking of all individuals and families for two calendar years
- Person-level and event-level annual data available since 1996

MEPS-HC data provides a useful basis for cross-package comparisons because variances can be estimated using both Taylor-series linearization and replicate weighting methods, unlike most complex sample survey data sets which recommend only one or the other. Presenting a step-by-step methodology with the MEPS-HC data set allows more convenient generalization to other data sets.

Data preparation

AHRQ provides all downloadable MEPS files in both ASCII (.txt) and SAS transport (.xpt and .ssp) formats, along with programming statements for SAS and SPSS. The **foreign** package imports SAS transport files with minimal effort:

```
> library(foreign)
> h105 <- read.xport("h105.ssp")
```

Reading larger data sets with the `read.xport` function may overburden system memory.[3] If a memory error occurs, SAS transport files can be converted to tab-delimited (.txt) files using the free SAS System Viewer® and then alternatively imported with the `read.table` function:

```
> h105 <- read.table("h105.dat", header=TRUE )
```

[1] http://www.cms.hhs.gov/NationalHealthExpendData/downloads/highlights.pdf
[2] http://www.meps.ahrq.gov/mepsweb/data_stats/download_data_files.jsp
[3] http://www.biostat.jhsph.edu/~rpeng/docs/R-large-tables.html

Should memory errors continue to occur due to a large number of variables in the data set, database-backed survey design objects allow R to access a table stored within a relational database using either DBI or ODBC.[4] This alternative does not conserve memory in data sets that are large due to record count. The table is accessed directly by the `svydesign` function (discussed later), through the addition of a `dbname` argument to specify the location and a `dbtype` argument to specify the database driver.

After successful import, unnecessary variables should be eliminated from the data frame to increase processing speed. The `select` argument to the `subset` command acts as the SAS *KEEP* option in a DATA step or the Stata *keep* command, by dropping all unspecified fields. Distinct subpopulation data frames could also be created with the `subset` command; however, observations should not be excluded until the design object has been created, in order to preserve the entire complex sample design.[5]

```
> #limit to specified fields
> meps06 <- subset(h105,
+   select = c(DUPERSID, PANEL, PERWT06F, VARSTR,
+     VARPSU, AGE06X, TOTEXP06, REGION06, RTHLTH53))

> #clear memory of full data frame
> rm(h105)
```

All other recoding and re-categorization should occur at this stage. The example below creates a categorical variable based on the age of the individual using the `cut` function to delineate the appropriate intervals.

```
> #recode linear to categorical
> # below 25 ; 25-34 ; 35-44 ; 45-54 ;
> # 55-64 ; 65-74 ; 75-84 ; 85+
> meps06[, "AGECAT"] <- cut(meps06[, "AGE06X"],
+   br = c(-1, 24 + 10*0:6, Inf), labels = 1:8)
```

Note that the `labels` argument can directly produce character strings rather than numeric categories. This example creates the same categories as above, with character labels.

```
> #alternate character labels
> meps06[, "cAGECAT"] <- cut(meps06[, "AGE06X"],
+   br = c(-1, 24+10*0:6, Inf),
+   labels = c('Below 25', '25-34', '35-44',
+     '45-54', '55-64', '65-74', '75-84', '85+'))
```

Once all needed variables have been created, inapplicable or invalid values should be converted to missing. This example combines the 'Excellent' and 'Very good' self-reported health status responses, then assigns incomplete records as missing values.[6]

```
> #combine ex (1) & vg health (2)
> # into the same response option
> meps06[, "RTHLTH53"] <- meps06[ ,"RTHLTH53"] - 1
> meps06[meps06[ ,"RTHLTH53"] == 0, "RTHLTH53"] <-
+   1

> #recode negatives to missing (NA)
> meps06[meps06[ , "RTHLTH53"] < 0, "RTHLTH53"] <-
+   NA
```

The addition of missing values to any variable within a data frame requires that subsequent analyses of the variable include the `na.rm = TRUE` argument in order to appropriately exclude those observations.

At the conclusion of all data transformations, all variables that one wishes to display in weighted column frequencies should be converted to a character string in a new variable. The rationale behind this conversion is discussed in the *Analysis* section.

```
> #char vars for region & health stat.
> meps06 <- transform(meps06,
+   cRTHLTH53 = as.character(RTHLTH53))
> meps06 <- transform(meps06,
+   cREGION06 = as.character(REGION06))
```

Survey design for Taylor-series linearization

The estimation weight (PERWT06F), stratum (VARSTR), and primary sampling unit (VARPSU) fields make up the MEPS survey design and allow for the Taylor-series linearization method of standard error computation. The **survey** package can then be used to create a complex sample survey design object for each data set. After creation, the complex survey design object replaces the R data frame as the target of all analysis functions.

```
> library(survey)

> #create main tsl survey design obj.
> meps.tsl.dsgn <- svydesign(id = ~ VARPSU,
+   strata = ~ VARSTR, weights = ~ PERWT06F,
+   data = meps06, nest = TRUE)
```

Survey design objects store (rather than point to) data sets at the time of design object creation; any modifications of the `meps06` data frame in subsequent steps require the above re-assignment of the design objects in order to see those changes in subsequent analysis commands.

[4] http://faculty.washington.edu/tlumley/survey/svy-dbi.html

[5] http://faculty.washington.edu/tlumley/survey/example-domain.html

[6] Original Self-Reported Health Status Values in MEPS: (1) Excellent; (2) Very good; (3) Good; (4) Fair; (5) Poor; (-9) Not Ascertained; (-8) Don't Know; (-7) Refused; (-1) Inapplicable.

Taylor-series linearization cross-package comparisons

Many health policy data sets with complex sample designs use the Taylor-series linearization method for standard error computation. After the appropriate survey design object has been created, point estimates can be obtained.

```
> #total adult expendure mean & se
> svymean(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #total nonelderly adult expenditure
> svymean(~ TOTEXP06 ,
+   subset(meps.tsl.dsgn, AGE06X > 17
+     & AGE06X < 65))
```

As seen in Table 1, R accurately replicates the estimates derived by each of the other statistical packages, SAS, Stata, and SUDAAN. The programming code for each of these other three packages matches the AHRQ recommendations found in their data documentation. [7]

Since policy research and evaluations often dictate that adults and children be studied separately, this paper will chiefly analyze the data set after restricting the sample to individuals aged 18 or above. Although used throughout this paper, note that the `subset` design object embedded in the functions above could be re-created as a pointer.

```
> x <- subset(meps.tsl.dsgn, AGE06X > 17)
```

Data preparation and survey design for replicate weighting

Note: Analysts solely interested in Taylor-series linearization can skip this section. Standard errors in MEPS-HC can be calculated without the Balanced Repeated Replication (BRR) technique; however, these survey design steps can be generalized to other health policy data sets that recommend replicate weighting, including the Consumer Expenditure Survey (CE) and the Survey of Income and Program Participation (SIPP). Replicate weighting designs provide the dual advantages of allowing quantile point estimate standard errors to be calculated with relative ease and of maintaining accurate variances when a subpopulation under examination has been so restricted as to contain a stratum with only one PSU.[8] The code below imports the MEPS-HC Replicates file and joins that file with the adult file and the non-elderly adult file.

```
> #load the half-sample data
> library(foreign)
> brr <- read.xport("h36b06.ssp")

> #merge half-sample with main file
> meps.brr <- merge(meps06, brr,
+   by.x = c("DUPERSID", "PANEL"),
+   by.y = c("DUPERSID", "PANEL"),
+   all=FALSE)
```

Note that if additional variables need to be removed from a data frame, the select argument captures all fields in between two fields separated by a colon - a shortcut particularly valuable for maintaining 64 replicate weight variables.

```
> meps.brr.sub <- subset(meps.brr,
+   select = c(DUPERSID, PANEL, PERWT06F,
+     VARSTR, VARPSU, AGE06X, AGECAT,
+     TOTEXP06, REGION06, cREGION06,
+     RTHLTH53, cRTHLTH5, BRR1:BRR64))
```

The survey replicate weight function `svrepdesign` allows the creation of a replicate weight-based survey design object. The `repweights` argument requires the identification of the 64 replicate weight fields in the data frame, obtained in the code below by isolating all fields containing the string `"BRR"` with the `grep` function. Half-sample replication techniques generally rely on the replicate weights equation BRR**X**wt = BRR**X** * 2 * PERWT06F, where **X** is 1:64 and each BRR**X** field is a binary variable containing an equal number of zeroes and ones that are randomly distributed across the records in the data set. If the series of replicate weights has already been computed to BRR**X**wt, the `combined.weights` argument should be set to `TRUE`. The downloadable MEPS BRR file contains only binary BRR**X** variables, indicating that R must take the extra step of computing the BRR**X**wt weights.[9]

```
> #create meps brr survey design obj.
> meps.brr.dsgn <-svrepdesign(data = meps.brr,
+   repweights =
+     meps.brr[ ,grep("^BRR", colnames(meps.brr))],
+   type = "BRR", combined.weights = FALSE,
+   weights = meps.brr$PERWT06F)
```

Replicate weighting cross-package comparisons

Note: Analysts solely interested in Taylor-series linearization can skip this section. Data sets used for health policy research often require the calculation of standard errors using replicate weights. Using the same `svymean` function, the point estimate can be obtained again by simply changing the survey design object.

```
> #brr method: total $ mean & se
> svymean(~ TOTEXP06,
+   subset(meps.brr.dsgn, AGE06X > 17))
```

[7] http://www.meps.ahrq.gov/mepsweb/survey_comp/standard_errors.jsp
[8] http://faculty.washington.edu/tlumley/survey/exmample-lonely.html
[9] http://www.meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-036BRR

Table 1: Comparison of Taylor-Series Linearization Standard Error Calculation Methods

		R	SAS	Stata	SUDAAN
All Adults	Code	`svymean(~TOTEXP06, subset( meps.tsl.dsgn, AGE06X > 17))`	`proc surveymeans data=meps.adult; stratum varstr; cluster varpsu; weight perwt06f; var totexp06;`	`svyset [pweight=PERWT06F], strata(VARSTR) psu(VARPSU) svy: mean TOTEXP06`	`proc descript data="adult" filetype=sasxport design=wr notsorted; nest varstr varpsu; weight perwt06f; var totexp06;`
	Mean	4009.1	4009.1	4009.1	4009.1
	SE	82.4	82.4	82.4	82.4
Non-Elderly Adults	Code	`svymean(~TOTEXP06, subset( meps.tsl.dsgn, AGE06X > 17 & AGE06X < 65))`	`proc surveymeans data=meps.adult; where age06x < 65; stratum varstr; cluster varpsu; weight perwt06f; var totexp06;`	`svyset [pweight=PERWT06F], strata(VARSTR) psu(VARPSU) svy: mean TOTEXP06 if AGE06X < 65`	`proc descript data = "adult" filetype=sasxport design=wr notsorted; nest varstr varpsu; weight perwt06f; var totexp06; subpopn age06x < 65;`
	Mean	3155.4	3155.4	3155.4	3155.4
	SE	76.2	76.2	76.2	76.2

As seen in Table 2, the survey design object created by R matches the Stata variance estimation precisely where the mean-squared error (MSE) option has *not* been selected; this estimate is derived from the sum of squares of the replicate estimates centered at the mean of those estimates. Stata's variance estimation using the MSE option and SUDAAN's built-in replicate weighting algorithm both center on the actual point estimate rather than the mean of the replicate estimates, resulting in a slightly larger confidence interval. The choice between these two techniques does not have any theoretical considerations, and the results are almost always similar. Though marginally different, standard errors computed with R are equally valid as those computed with SUDAAN or with Stata's MSE method. Two statistical tests conducted using the standard errors from the different packages will almost always produce a numerically different but substantively compatible result.

Comparison code for SAS was not included in Table 2 because the base package requires the calculation of replicate weighting estimates with user-written code.

Analysis

The core functions included in R's **survey** package allow the rapid estimation and cross-tabulation of means, distributions, frequencies, and quantiles. Each of the discussed operations also produces standard errors, except for the quantile calculations using Taylor-series linearization design objects.

Unweighted frequencies

Before conducting analysis on the complex survey design object, simple unweighted frequencies can be produced using the `xtabs` function. Unweighted frequencies only assure that sufficient cell sizes exist, meaning that the complex design does not need to be accounted for. Therefore, this function refers to the source data set, rather than the design object.

```
> #calculate unweighted cell size
> xtabs( ~ RTHLTH53 + REGION06,
+   subset(meps06, AGE06X > 7))
```

Distributions

As previously shown, the `svymean` function calculates mean values for any numeric variable. When given character variables, however, `svymean` computes percentages of the total. Notice the presence of the `na.rm = TRUE` argument to eliminate missing values from the results.

```
> #percent distribution w/ svymean
> #determine % adults in each state
> # (ex. poor health 3.26%)
> a <- svymean(~ cRTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17),
+     na.rm = TRUE)
> a
```

Although this paper focuses on data manipulation and analysis, R includes powerful formatting and presentation tools. Individuals interested in converting simple tables — like the one produced by the code below — into more advanced graphics should research the R packages **graphics**, **lattice**, **grid**, **grDevices**, as well as LaTeX converters like **Hmisc** and **xtable**.

```
> #label & format health status
> ftable(a,
+   list(c("Excellent / Very good",
+     "Good", "Fair", "Poor")))
```

Variables not converted to character before the creation of the design object can be hastily converted using the `as.character` function within the `svymean` command. Also notice the absence of the `na.rm` argument, since missing values were not assigned for any cases of the *REGION06* variable.

```
# % adults living in census regions
# (ex. northeastern 18.64%)
> svymean(~ as.character(REGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17))
```

Weighted frequencies and totals

The `svytotal` and `svytable` functions can be used to calculate both weighted aggregate values and population estimates.

```
> #total 2006 health expenditures
> #all non-institutionalized adults
> # ($892 billion)
> svytotal(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #num adults in census regions
> # (ex. northeastern 41.47 million)
> svytable(~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #alternate method
> svytotal(~ as.character(REGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17))
```

Medians and quantiles

The `svyquantile` function calculates each of the percentiles specified in the `quantile` argument. For example, a value of 0.5 generates the median. However, any number of quantiles can be produced with by specifying a vector of values.

```
> #median 2006 health expenditure
> svyquantile(~ TOTEXP06,
```

Table 2: Comparison of Balanced Repeated Replication Standard Error Calculation Methods

		R	Stata	Stata (MSE)	SUDAAN
All Adults	Code	`svymean(~TOTEXP06, subset( meps.brr.dsgn, AGE06X > 17))`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) svy: mean TOTEXP06`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) mse svy: mean TOTEXP06`	`proc descript data = "adult" filetype=sasxport design=brr notsorted; repwgt BRR1 - BRR64 ; weight perwt06f; var totexp06;`
	Mean	4009.1	4009.1	4009.1	4009.1
	SE	83.0	83.0	83.6	83.6
Non-Elderly Adults	Code	`svymean(~TOTEXP06, subset( meps.brr.dsgn, AGE06X > 17 & AGE06X < 65))`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) svy: mean TOTEXP06 if AGE06X < 65`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) mse svy: mean TOTEXP06 if AGE06X < 65`	`proc descript data = "adult" filetype=sasxport design=brr notsorted; repwgt BRR1 - BRR64 ; weight perwt06f; var totexp06; subpopn age06x < 65;`
	Mean	3155.4	3155.4	3155.4	3155.4
	SE	77.0	77.0	77.5	77.5

```
+   subset(meps.tsl.dsgn, AGE06X > 17), 0.5)

> #calculate expenditure percentiles
> svyquantile(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   c(0.1, 0.25, 0.5, 0.75, 0.9))
```

Although there are a number of other methods for constructing an interval estimate, the Taylor-series linearization design object will not produce standard errors in `svyquantile` since the CDF is not differentiable.[10] Median standard errors can be obtained normally if a replicate weighting design object exists for the data.

```
> #brr method: median health $
> svyquantile(~ TOTEXP06,
+   subset(meps.brr.dsgn, AGE06X > 17), 0.5)
```

If replicate weights do not exist for the data set, a jackknife replication design can be created by converting the Taylor-series linearization design object.

```
> #convert tsl to jackknife design
> meps.jackknife.dsgn <-
+   as.svrepdesign(meps.tsl.dsgn, "auto")

> #jackknife method: median health $
> svyquantile(~ TOTEXP06,
+   subset(meps.jackknife.dsgn, AGE06X > 17), 0.5)
```

Although this jackknife technique might produce a ballpark standard error statistic, recommended quantile standard error calculations can vary depending on the data set and analysis method. Some textbooks simply estimate median standard error as 25% larger than the mean standard error.[11] Therefore, technical documentation and/or data support staff should be consulted before relying on this survey design conversion.

Crosstabulation of totals, means, and quantiles

The `svyby` command repeats a specified function, iterating across the set of distinct values in another factor variable also within the data frame. This function is comparable to `lapply`, which runs across individual elements contained in an external vector. Rather than running the indicated function on the entire data set, `svyby` analyzes discrete subpopulations.

```
> #health spending by census region
> # (ex. northeast $173 billion)
> svyby(~ TOTEXP06, ~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svytotal)

> #regional mean spending on health
> # (ex. northeast $4,171)
> svyby(~ TOTEXP06, ~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #stratified by health status
> # (ex. northeast poor hlth $18,038)
> svyby(~ TOTEXP06, ~ REGION06 + RTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #top quartile by health status
> # (ex. poor health $19,130)
> svyby(~ TOTEXP06, ~ RTHLTH53,
+   design = subset(meps.tsl.dsgn, AGE06X > 17),
+   FUN = svyquantile, quantiles = 0.75, ci = TRUE)
```

Crosstabulation of distributions

In addition to the straightforward distributions presented using the `svymean` function, the `svyby` function allows row and column percents to be calculated using any desired factor variable(s).

```
> #dist of hlth stat by census region
> svyby(~ cREGION06, ~cRTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #dist of census region by hlth stat
> svyby(~ cRTHLTH53, ~cREGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean,
+   na.rm = TRUE)
```

Unlike row and column percents, table percents do not require the `svyby` command and can be calculated using the `interaction` function. This approach is principally useful when one wishes to examine a sub-subpopulation in relation to the entire population, rather than in relation to its parent subpopulation.

```
> #total % in each hlth stat & region
> svymean(~ interaction(cRTHLTH53, cREGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   na.rm = TRUE)

> #repeat above with weighted counts
> svytotal(~ interaction(cRTHLTH53, cREGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   na.rm = TRUE)
```

Statistical tests

Though most individual estimates are more easily compared using a general difference formula, the `svyttest` function can be used to conduct quick two-sample t-tests on any binary value that can be generated in the data set. For example, to assess statistical differences in health spending between adults in a particular census region of the United States and adults living in all other regions, one can rapidly create a binary variable to test for spending differences.

[10] http://faculty.washington.edu/tlumley/survey/html/svyquantile.html
[11] http://davidmlane.com/hyperstat/A106993.html

```
> #spending in northeast vs. others
> svyttest(TOTEXP06 ~ (cREGION06 == 1),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))

> #spending in midwest vs. others
> svyttest(TOTEXP06 ~ (cREGION06 == 2),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))
```

Using this method, any Boolean expression can be tested for differences between groups. The examples below show that adults aged 45 to 54 do not have significantly higher or lower spending than the set of all other adults, while pre-retirees (aged 55 to 64r) do spend significantly more than all other adults.

```
> #test 45-54 vs. all others
> svyttest(TOTEXP06 ~ (AGECAT == 4),
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #test 55-64 vs. all others
> svyttest(TOTEXP06 ~ (cAGECAT == '55-64'),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))
```

Conclusion

This article has outlined the capacity to prepare and accurately analyze health policy data with complex survey designs using R. By highlighting the necessary steps to transition from the proprietary statistical package code provided in the technical documentation of the MEPS-HC data, this article equips R users with the conversion knowledge required to conduct important research on health policy data sets. The data manipulation and analysis techniques proven to work on MEPS can be generalized to the majority of available health policy data sets.

As a secondary goal behind contributing to the health policy analyst's toolkit, this article has laid the groundwork for health-care survey administration organizations to follow the lead of the National Center for Health Statistics by including instructions and example code for R in future versions of their technical documentation.[12]

Version notes

The calculations in this paper were performed with R version 2.9.0, using the R survey package version 3.16. Statistical package comparisons were calculated using SAS version 9.1.3 Service Pack 2, Stata/MP version 9.2, and SUDAAN release 10.0.0.

Acknowledgements

I gratefully acknowledge Dr. Thomas Lumley, Professor of Biostatistics at the University of Washington, Seattle and author of the R survey package for assistance with the replication design methodology, and Dr. Roger Peng, Professor of Biostatistics at Johns Hopkins School of Public Health for code editing.

Anthony Damico
Statistical Analyst
Kaiser Family Foundation
adamico@kff.org

[12] ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHIS/2008/srvydesc.pdf

Rattle: A Data Mining GUI for R

by Graham J Williams

Abstract: Data mining delivers insights, patterns, and descriptive and predictive models from the large amounts of data available today in many organisations. The data miner draws heavily on methodologies, techniques and algorithms from statistics, machine learning, and computer science. R increasingly provides a powerful platform for data mining. However, scripting and programming is sometimes a challenge for data analysts moving into data mining. The Rattle package provides a graphical user interface specifically for data mining using R. It also provides a stepping stone toward using R as a programming language for data analysis.

Introduction

Data mining combines concepts, tools, and algorithms from machine learning and statistics for the analysis of very large datasets, so as to gain insights, understanding, and actionable knowledge.

Closed source data mining products have facilitated the uptake of data mining in many organisations. These products offer off-the-shelf ease-of-use that makes them attractive to the many new data miners in a market place desperately seeking high levels of analytical skills.

R is ideally suited to the many challenging tasks associated with data mining. R offers a breadth and depth in statistical computing beyond what is available in commercial closed source products. Yet R remains, primarily, a programming language for the highly skilled statistician, and out of the reach of many.

Rattle (the R Analytical Tool To Learn Easily) is a graphical data mining application written in and providing a pathway into R (Williams, 2009b). It has been developed specifically to ease the transition from basic data mining, as necessarily offered by GUIs, to sophisticated data analyses using a powerful statistical language.

Rattle brings together a multitude of R packages that are essential for the data miner but often not easy for the novice to use. An understanding of R is not required in order to get started with Rattle—this will gradually grow as we add sophistication to our data mining. Rattle's user interface provides an entree into the power of R as a data mining tool.

Rattle is used for teaching data mining at numerous universities and is in daily use by consultants and data mining teams world wide. It is also available as a product within Information Builders' WebFocus business intelligence suite as RStat.

Rattle is one of several open source data mining tools (Chen et al., 2007). Many of these tools are also directly available within R (and hence Rattle) through packages like **RWeka** (Hornik et al., 2009) and **arules** (Hahsler et al., 2005).

Implementation

Rattle uses the Gnome graphical user interface as provided through the **RGtk2** package (Lawrence and Lang, 2006). It runs under various operating systems, including GNU/Linux, Macintosh OS/X, and MS/Windows.

The GUI itself has been developed using the Glade interactive interface builder. This produces a programming-language-independent XML description of the layout of the widgets that make up the user interface. The XML file is then simply loaded by an application and the GUI is rendered!

Glade allows the developer to freely choose to implement the functionality (i.e., the widget callbacks) in a programming language of choice, and for Rattle that is R. It is interesting to note that the first implementation of Rattle actually used Python for implementing the callbacks and R for the statistics, using **rpy**. The release of **RGtk2** allowed the interface elements of Rattle to be written directly in R so that Rattle is a fully R-based application.

Underneath, Rattle relies upon an extensive collection of R packages. This is a testament to the power of R—it delivers a breadth and depth of statistical analysis that is hard to find anywhere else. Some of the packages underlying Rattle include **ada**, **arules**, **doBy**, **ellipse**, **fBasics**, **fpc**, **gplots**, **Hmisc**, **kernlab**, **mice**, **network**, **party**, **playwith**, **pmml**, **randomForest**, **reshape**, **rggobi**, **RGtk2**, **ROCR**, **RODBC**, and **rpart**. These packages are all available from the Comprehensive R Archive Network (CRAN). If a package is not installed but we ask through Rattle for some functionality provided by that package, Rattle will popup a message indicating that the package needs to be installed.

Rattle is not only an interface though. Additional functionality that is desired by a data miner has been written for use in Rattle, and is available from the **rattle** package without using the Rattle GUI. The **pmml** package (Guazzelli et al., 2009) is an offshoot of the development of Rattle and supports the export of models from Rattle using the open standard XML based PMML, or Predictive Model Markup Language (Data Mining Group, 2008). Models exported from R in this way can be imported into tools like the ADAPA decision engine running on cloud computers, Teradata's Warehouse Miner for deployment as SQL over a very large database, and Information Builder's WebFocus which handles

data sourcing, preparation, and reporting, and is able to transform Rattle generated PMML models into C code to run on over 30 platforms.

Installation

The Gnome and Glade libraries need to be installed (separately to R) to run Rattle. On GNU/Linux and Mac/OSX this is usually a simple package installation. Specifically, for Debian or Ubuntu we install packages like **gnome** and **glade-3**. For MS/Windows the self-installing libraries can be obtained from `http://downloads.sourceforge.net/gladewin32`. Full instructions are available from `http://rattle.togaware.com`.

After installing the required libraries be sure to restart the R console to ensure R can find the new libraries.

Assuming R is installed we can then install the **RGtk2** and **rattle** packages with:

```
> install.packages("RGtk2")
> install.packages("rattle")
```

Once installed we simply start Rattle by loading the **rattle** package and then evaluating the `rattle()` function:

```
> library(rattle)

Rattle: Graphical interface for data mining in R.
Version 2.5.0 Copyright (C) 2006-2009 Togaware.
Type 'rattle()' to shake, rattle, & roll your data.

> rattle()
```

Rattle will pop up a window similar to that in Figure 1.

The latest development version of Rattle is always available from Togaware:

```
> install.packages("rattle",
+     repos = "http://rattle.togaware.com")
```

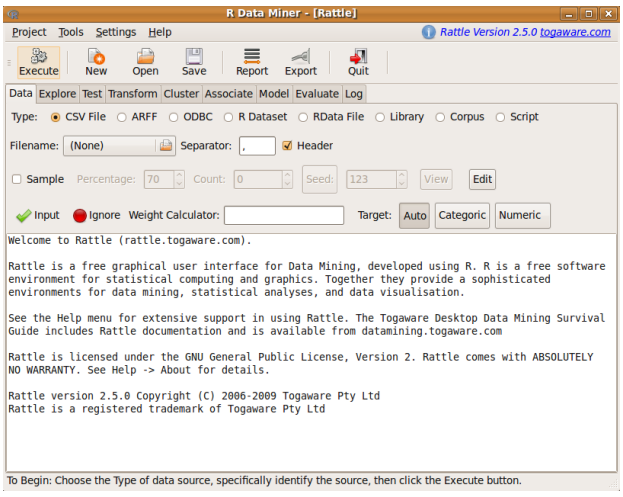

Figure 1: Rattle's introductory screen.

Data Mining

Rattle specifically uses a simple tab-based concept for the user interface (Figure 1), capturing a work flow through the data mining process with a tab for each stage. A typical work flow progresses from the left most tab (the **Data** tab) to the right most tab (the **Log** tab). For any tab the idea is for the user to configure the available options and then to click the **Execute** button (or `F2`) to perform the appropriate task. The status bar at the base of the window will indicate when the action is complete.

We can illustrate a very simple, if unrealistic, run through Rattle to build a data mining model with just four mouse clicks. Start up R, load the **rattle** package, and issue the `rattle()` command. Then:

1. Click on the **Execute** button;
2. Click on **Yes** within the resulting popup;
3. Click on the **Model** tab;
4. Click on the **Execute** button.

Now we have a decision tree built from a sample classification dataset.

With only one or two more clicks, alternative models can be built. A few more clicks will have an evaluation chart displayed to compare the performance of the models constructed. Then a click or two more will have the models applied to score a new dataset.

Of course, there is much more to modelling and data mining than simply building a tree model. This simple example provides a flavour of the interface provided by Rattle.

The common work flow for a data mining project can be summarised as:

1. Load a **Data**set and select variables;
2. **Explore** the data to understand distributions;
3. **Test** distributions;
4. **Transform** the data to suit the modelling;
5. Build **Model**s;
6. **Evaluate** models and score datasets;
7. Review the **Log** of the data mining process.

The underlying R code, constructed and executed by Rattle, is recorded in the **Log** tab, together with instructive comments. This allows the user to review the actual R commands. The R code snippets can also be copied as text (or **Export**ed to file) from the **Log** tab and pasted into the R console and executed. This allows Rattle to be deployed for basic tasks, yet still access the full power of R as needed (e.g., to fine-tune modelling options that are not exposed in the interface).

The use of Sweave (Leisch, 2002) to allow LaTeX markup as the format of the contents of the log is

experimental but will introduce the concept of literate data mining. The data miner will document their activity, as they proceed through Rattle, by editing the log which is also automatically populated as the modelling proceeds. Simple and automatic processing can then turn the log into a formatted report that also embodies the actual code, which may also be run so as to replicate the activity.

Using the related Tangle processor allows the log to be exported as an R script file, to record the actions taken. The script can then be independently run at a later time (or pasted into the R console).

Repeatability and reproducibility are important in both scientific research and commercial practice.

Data

If no dataset has been supplied to Rattle and we click the **Execute** button (e.g., startup Rattle and immediately click **Execute**) we are given the option to load one of Rattle's sample datasets from a CSV file.

Rattle can load data from various sources. It directly supports CSV (comma separated data), TXT (tab separated data), ARFF (a common data mining dataset format (Witten and Frank, 2005) which adds type information to a CSV file), and ODBC connections (allowing connection to many data sources including MySQL, SQLite, Postgress, MS/Excel, MS/Access, SQL Server, Oracle, IBM DB2, Netezza, and Teradata). R data frames attached to the current R session, and datasets available from the packages installed in the R libraries, are also available through the Rattle interface.

To explore the use of Rattle as a data mining tool we consider the sample `audit` dataset provided by the **rattle** package. The data is artificial, but reflects a real world dataset used for reviewing the outcomes of historic financial audits. Picture, for example, a revenue authority collecting taxes based on information supplied by the tax payer. Thousands of random audits might be performed and the outcomes indicate whether an adjustment to the supplied information was required, resulting in a change to the taxpayer's liability.

The audit dataset is supplied as both an R dataset and as a CSV file. The dataset consists of 2,000 fictional tax payers who have been audited for tax compliance. For each case an outcome after the audit is recorded (whether the financial claims had to be adjusted or not). The actual dollar amount of adjustment that resulted is also recorded (noting that adjustments can go in either direction).

The audit dataset contains 13 variables (or columns), with the first being a unique client identifier.

When loading data into Rattle certain special prefixes to variable names can be used to identify default variable roles. For example, if the variable name starts with 'ID_' then the variable is marked as having a role as an identifier. Other prefixes include 'IGNORE_', 'RISK_', 'IMP_' (for imputed) and 'TARGET_'. Examples from the audit data include `IGNORE_Accounts` and `TARGET_Adjusted`.

The CSV option of the **Data** tab provides the simplest approach to loading data into Rattle. If the **Data** tab is **Execute**d with no CSV file name specified then Rattle offers the option to load a sample dataset. Clicking on the **Filename** box will then list other available sample datasets, including 'audit.csv'.

Once Rattle loads a dataset the text window will contain the list of available variables and their default roles (as in Figure 2).

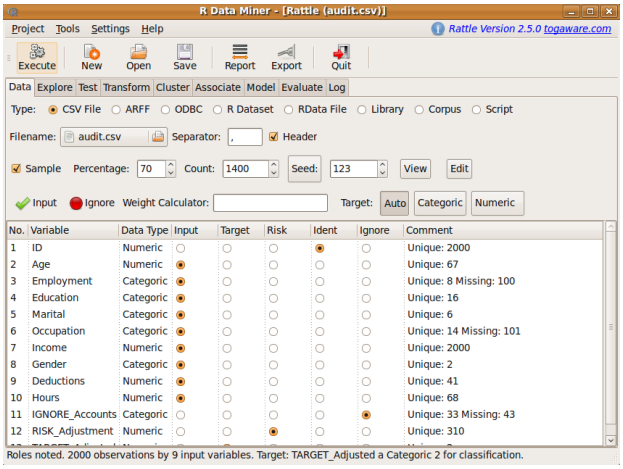

Figure 2: Rattle's variable roles screen.

By default, most variables have a role of **Input** for modelling. We may want to identify one variable as the **Target** variable, and optionally identify another variable as a **Risk** variable (which is a measure of the size of the "targets"). Other roles include **Ident** and **Ignore**.

Rattle uses simple heuristics to guess at roles, particularly for the target and ignored variables. For, example, any numeric variable that has a unique value for each observation is automatically identified as an identifier.

Rattle will, by default, partition the dataset into a training and a test dataset. This kind of sampling is useful for exploratory purposes when the data is quite large. Its primary purpose, though, is to select a 70% sample for training of models, providing a 30% set for testing.

Explore

Exploratory data analysis is important in understanding our data. The **Explore** tab provides numerous numeric and graphic tools for exploring data. Once again, there is a considerable reliance on many other R packages.

Summary

The **Summary** option uses R's `summary` command to provide a basic univariate summary. This is augmented with the `contents` and `describe` commands from the **Hmisc** package (Harrell, 2009). Extended summaries include additional statistics provided by the **fBasics** package (Wuertz, 2009), kurtosis and skewness, as well as a summary of missing values using the missing value functionality from the **mice** package (van Buuren and Groothuis-Oudshoorn, 2009).

Distributions

The **Distributions** option provides access to numerous plot types. It is always a good idea to review the distributions of the values of each of the variables before we consider data mining. While the above summaries help, the visual explorations can often be quite revealing (Cook and Swayne, 2007).

A vast array of tools are available within R for presenting data visually and the topic is covered in detail in many books including Cleveland (1993). Rattle provides a simple interface to the underlying functionality in R for drawing some common plots. The current implementation primarily relies on the base graphics provided by R, but may migrate to the more sophisticated **lattice** (Sarkar, 2008) or **ggplot2** (Wickham, 2009).

Some of the canned plots are illustrated in Figure 3. Clockwise we can see a box plot, a histogram, a mosaic plot, and a Benford's Law plot. Having identified a target variable (in the **Data** tab) the plots include the distributions for each subset of observations associated with each value of the target variable, wherever this makes sense to do so.

GGobi and Latticist

Rattle provides access to two sophisticated tools for interactive graphical data analysis: GGobi and Latticist.

The GGobi (Cook and Swayne, 2007) visualisation tool is accessed through the **rggobi** package (Wickham et al., 2008). GGobi will need to be installed on the system separately, and runs under GNU/Linux, OS/X, and MS/Windows. It is available for download from `http://www.ggobi.org/`.

Ggobi is useful for exploring high-dimensional data through highly dynamic and interactive graphics, especially with tours, scatterplots, barcharts and parallel coordinate plots. The plots are interactive and linked with brushing and identification. The available functionality is extensive, and supports panning, zooming and rotations.

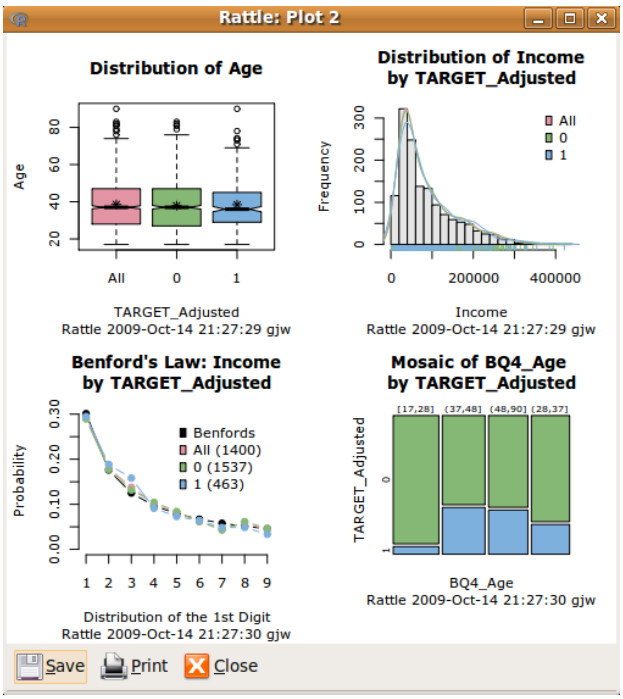

Figure 3: Exploring variable distributions.

Figure 4 displays a scatterplot of Age versus Income (left) and a scatterplot matrix across four variables at the one time (right). Brushing is used to distinguish the class of each observation.

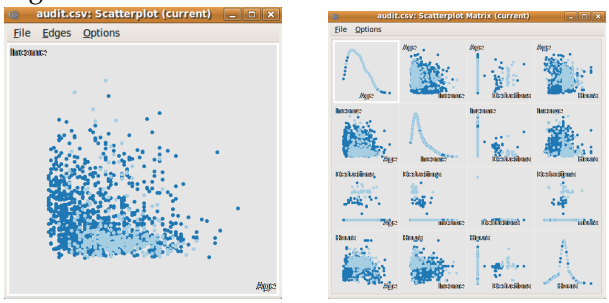

Figure 4: Example of GGobi using **rggobi** to connect.

A more recent addition to the R suite of packages are the **latticist** and **playwith** packages (Andrews, 2008) which employ **lattice** graphics within a graphical interface to interactively explore data. The tool supports various plots, data selection and subsetting, and support for brushing and annotations. Figure 5 illustrates the default display when initiated from Rattle.

Test

The **Test** tab provides access to a number of parametric and non-parametric statistical tests of distributions. This more recent addition to Rattle continues to receive attention (and hence will change over time). In the context of data mining often applied to the binary classification problem, the current tests are primarily two sample statistical tests.

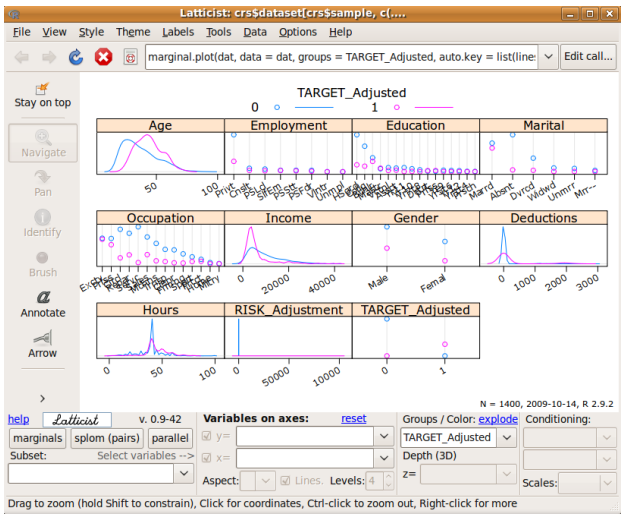

Figure 5: Latticist displaying the audit data.

Tests of data distribution include the Kolomogorov-Smirnov and Wilcoxon Signed Rank tests. For testing the location of the average the T-test and Wilcoxon Rank-Sum tests are provided. The F-test and Pearson's correlation are also available.

Transform

Cleaning data and creating new features (derived variables) occupies much time for a data miner. There are many approaches to data cleaning, and a programming language like R supports them all. Rattle's **Transform** tab (Figure 6) provides a number of the common options for transforming, including rescaling, skewness reduction, imputing missing values, turning numeric variables into categorical variables, and vice versa, dealing with outliers, and removing variables or observations with missing values. We review a number of the transforms here.

Rescale

The **Rescale** option offers a number of rescaling operations, using the `scale` command from base and the `rescaler` command from the **reshape** package (Wickham, 2007). Rescalings include recentering and scaling around zero (**Recenter**), scaling to 0–1 (**Scale [0,1]**), converting to a rank ordering (**Rank**), robust rescaling around zero using the median (**-Median/MAD**), and rescaling based on groups in the data.

For any transformation the original variable is not modified. A new variable is created with a prefix added to the variable's name to indicate the transformation. The prefixes include '`RRC_`', '`R01_`', '`RRK_`', '`RMD_`', and '`RBG_`', respectively.

The effect of the rescaling can be examined using the **Explore** tab (Figure 7). Notice that Rattle overlays bar charts with a density plot, by default.

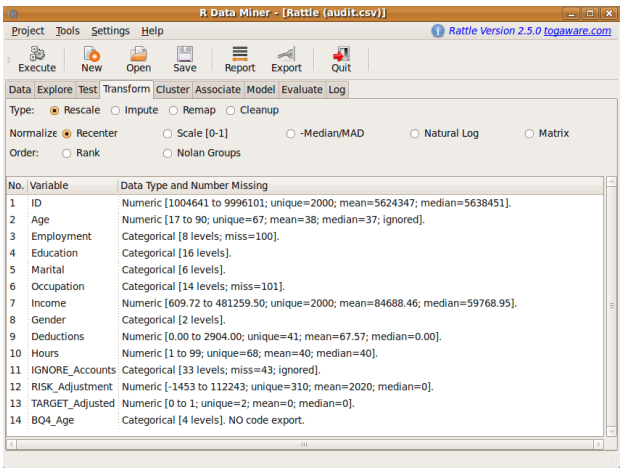

Figure 6: Transform options.

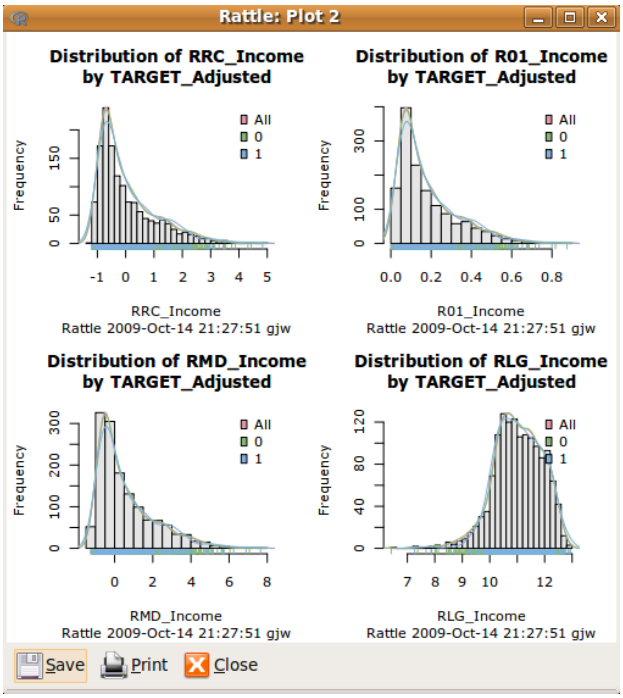

Figure 7: Four rescaled versions of `Income`.

Impute

Imputation of missing values is a tricky topic and should only be done with a good understanding of the data. Often, observational data (as distinct from experimental data) will contain missing values, and this can cause a problem for data mining algorithms. For example, the **Forest** option (using **randomForest**) silently removes any observation with any missing value! For datasets with a very large number of variables, and a reasonable number of missing values, this may well result in a small, unrepresentative dataset, or even no data at all!

There are many types of imputations possible, only some of which are directly available in Rattle. Further, Rattle does not (yet) support multiple imputation. The pattern of missing values can be viewed

using the **Show Missing** check button of the **Summary** option of the **Explore** tab.

The simplest, and least recommended, of imputations involves replacing all missing values for a variable with a single value. This makes most sense when we know that the missing values actually indicate that the value is "0" rather than unknown. For example, in a taxation context, if a tax payer does not provide a value for a specific type of deduction, then we might assume that they intend it to be zero. Similarly, if the number of children in a family is not recorded, it could be a reasonable assumption that it is zero (but it might equally well mean that the number is just unknown).

A common, if generally unsatisfactory, choice for missing values that are known not to be zero is to use some "central" value of the variable. This is often the mean, median, or mode. We might choose to use the mean, for example, if the variable is otherwise normally distributed (and in particular has little skewness). If the data does exhibit some skewness though (e.g., there are a small number of very large values) then the median might be a better choice.

Be wary of any imputation performed. It is, after all, inventing new data! Future development of Rattle may provide more support with model based imputation through packages like **Amelia** (Honaker et al., 2009).

Remap

The **Remap** option provides numerous re-mapping operations, including binning, log transforms, ratios, and mapping categorical variables into indicator variables for the situation where a model builder requires numeric data. Rattle provides options to use **Quantile** binning, **KMeans** binning, and **Equal Width** binning. For each option the default number of bins is 4 but we can change this to suit our needs. The generated variables are prefixed with either '`BQn_`', '`BKn_`', and '`BEn_`' respectively, with '`n`' replaced with the number of bins. Thus, we can create multiple binnings for any variable.

There are also options to **Join Categorics**—a convenient way to stratify the dataset, based on multiple categoric variables. A **Log** transform is also available.

Model

Data mining algorithms are often described as being either descriptive or predictive. Rattle currently supports the two common descriptive or unsupervised approaches to model building: cluster analysis and association analysis. A variety of predictive model builders are supported: decision trees, boosting, random forests, support vector machines, generalised linear models, and neural networks.

Predictive modelling, and generally the task of classification, is at the heart of data mining. Rattle originally focused on the common data mining task of binary (or two class) classification but now supports multinomial classification and regression, as well as descriptive models.

Rattle provides a straight-forward interface to a collection of descriptive and predictive model builders available in R. For each, a simple collection of tuning parameters is exposed through the graphical interface. Where possible, Rattle attempts to present good default values (often the same defaults as selected by the author of the respective package) to allow the user to simply build a model with no or little tuning. This may not always be the right approach, but is certainly a reasonable place to start.

We will review modelling within Rattle through decision trees and random forests.

Decision Trees

One of the classic machine learning techniques, widely deployed in data mining, is decision tree induction Quinlan (1986). Using a simple algorithm and a simple tree structure to represent the model, the approach has proven to be very effective. Underneath, the **rpart** (Therneau et al., 2009) and **party** (Hothorn et al., 2006) packages are called upon to do the work. Figure 8 shows the **Model** tab with the results of building a decision tree displayed textually (the usual output from the `summary` command for an "`rpart`" object).

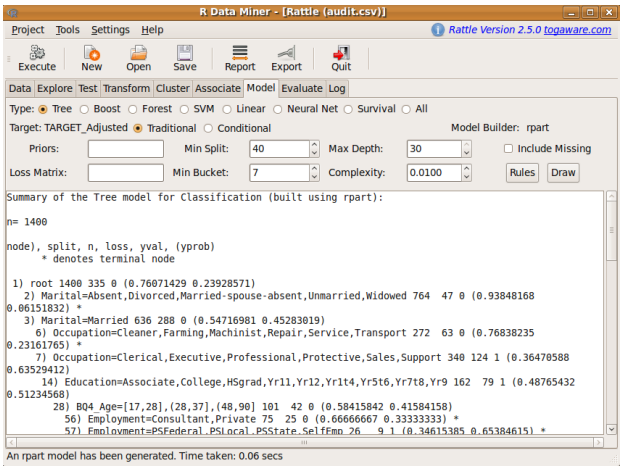

Figure 8: Building a decision tree.

Rattle adds value to the basic **rpart** functionality with additional displays of the decision tree, as in Figure 9, and the conversion of the decision tree into a list of rules (using the **Draw** and **Rules** buttons respectively).

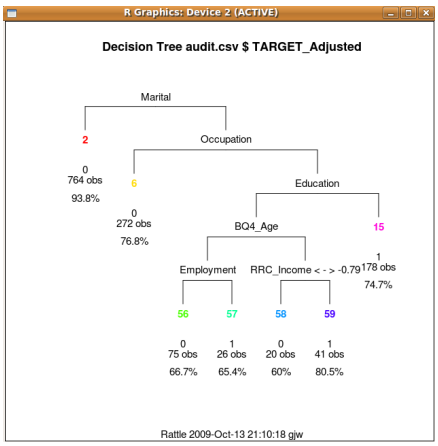

Figure 9: Rattle's display of a decision tree.

Ensemble

The ensemble approach has gained a lot of interest lately. Early work (Williams, 1988) experimented with the idea of combining a collection of decision trees. The results there indicated the benefit of building multiple trees and combining them into a single model, as an ensemble.

Recent developments continue to demonstrate the effectiveness of ensembles in data mining through the use of the boosting and random forest algorithms. Both are supported in rattle and we consider just the random forest here.

Random Forests

A random forest (Breiman, 2001) develops an ensemble of decision trees. Random forests are often used when we have very large training datasets and a very large number of input variables (hundreds or even thousands of input variables). A random forest model is typically made up of tens or hundreds of decision trees, each built using a random sample of the dataset, and whilst building any one tree, a random sample of the variables is considered at each node.

The random sampling of both the data and the variables ensures that even building 500 decision trees can be efficient. It also reputably delivers considerable robustness to noise, outliers, and overfitting, when compared to a single tree classifier.

Rattle uses the **randomForest** package (Liaw and Weiner, 2002) to build a forest of trees. This is an interface to the original random forest code from the original developers of the technique. Consequently though, the resulting trees have a different structure to standard `"rpart"` trees, and so some of the same tree visualisations are not readily available. Rattle can list all of the rules generated for a random forest, if required. For complex problems this can be a very long list indeed (thousands of rules).

The **Forest** option can also display a plot of relative variable importance. This provides insight into which variables play the most important role in predicting the class outputs. The **Importance** button will display two plots showing alternative measures of the relative importance of the variables in our dataset in relation to predicting the class.

Building All Models and Tuning

Empirically, the different model builders often produce models that perform similarly, in terms of misclassification rates. Thus, it is quite instructive to use all of the model builders over the same dataset. The **All** option will build one model for each of the different model builders.

We can review the performance of each of the models built and choose that which best suits our needs. In choosing a single model we may not necessarily choose the most accurate model. Other factors can come into play. For example, if the simple decision tree is almost as accurate as the 500 trees in the random forest ensemble, then we may not want to step up to the complexity of the random forest for deployment.

An effective alternative, where explanations are not required, and accuracy is desired, is to build a model of each type and to then build an ensemble that is a linear combination of these models.

Evaluate

Rattle provides a standard collection of tools for evaluating and comparing the performance of models. This includes the error matrix (or confusion table), lift charts, ROC curves, and Cost Curves, using, for example, the **ROCR** package (Sing et al., 2009). Figure 10 shows the options.

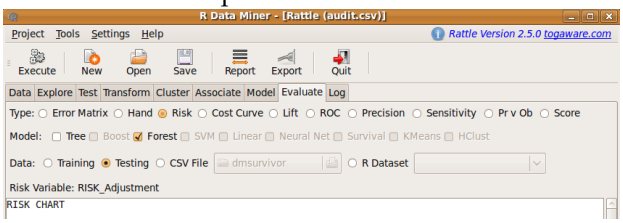

Figure 10: Options for Evaluation.

A cumulative variation of the ROC curve is implemented in Rattle as Risk charts (Figure 11). Risk charts are particularly suited to binary classification tasks, which are common in data mining. The aim is to efficiently display an easily understood measure of the performance of the model with respect to resources available. Such charts have been found to be more readily explainable to decision-making executives.

A risk chart is particularly useful in the context of the `audit` dataset, and for risk analysis tasks in general. The audit dataset has a two class target variable

as well as a so-called *risk* variable, which is a measure of the size of the risk associated with each observation. Observations that have no adjustment following an audit (i.e., clients who have supplied the correct information) will of course have a risk of zero associated with them. Observations that do have an adjustment will usually have a risk associated with them, and for convenience we simply identify the value of the adjustment as the magnitude of the risk.

Rattle uses the idea of a risk chart to evaluate the performance of a model in the context of risk analysis.

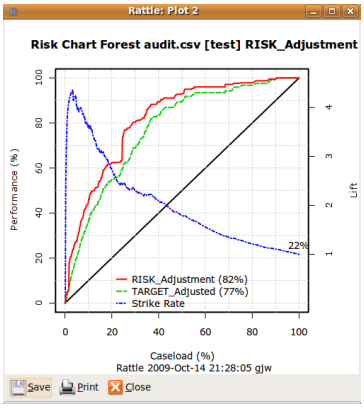

Figure 11: A simple cumulative risk chart.

A risk chart plots performance against caseload. Suppose we had a population of just 100 observations (or audit cases). The case load is the percentage of these cases that we will actually ask our auditors to process. The remaining cases will not be considered any further, expecting them to be low risk, and hence, with limited resources, not requiring any action.

The decision as to what percentage of cases are actioned corresponds to the x-axis of the risk chart—the caseload. A 100% caseload indicates that we will action all audit cases. A 25% caseload indicates that we will action just one quarter of all cases.

The performance is the percentage of the total number of cases that required an adjustment (or the total risk—both are plotted if a risk variable is identified) that might be covered in the population that we action.

The risk chart allows the trade-off between resources and risk to be visualised.

Model Deployment

Once we have decided upon a model that represents acceptable improvement to our business processes we are faced with deployment. Deployment can range from running the model ad hoc, to a fully automated and carefully governed deployment environment. We discuss some of the issues here and explore how Rattle supports deployment.

Scripting R

The simplest approach to deployment is to apply the model to a new dataset. This is often referred to as *scoring*. In the context of R this is nothing more than using the `predict` function.

Rattle's evaluation tab supports scoring with the **Score** option. There are further options to score the training dataset, the test dataset, or data loaded from a CSV file (which must contain the exact same variables). Any number of models can be selected, and the results are written to a CSV file.

Scoring is often performed some time after the model is built. In this case the model needs to be saved for later use. The concept of a Rattle project is useful in such a circumstance. The current state of Rattle (including the actual data and models built during a session) can be saved to a project, and later loaded into a new instance of Rattle (running on the same host or even a different host and operating system). A new dataset can then be scored using the saved model.

Underneath, saving/loading a Rattle project requires no more than using the `save` and `load` commands of R to create a binary representation of the R objects, and saving them to file. A Rattle project can get quite large, particularly with large datasets.

Larger files take longer to load, and for deploying a model it is often not necessary to keep the original data. So as we get serious about deployment we might save just the model we wish to deploy. This is done using the `save` function and knowing a little bit about the internals of Rattle (but no more than what is exposed through the **Log** tab).

The approach, saving a randomForest model, might be:

```
> myrf <- crs$rf
> save(myrf, file = "model01_090501.RData")
```

We can then load the model at a later time and apply the model (using a script based on the commands shown in the Rattle **Log** tab) to a new dataset:

```
> library(randomForest)
> (load("model01_090501.RData"))

[1] "myrf"

> dataset <- read.csv("cases_090601.csv")
> pr <- predict(myrf, dataset,
+     type = "prob")[, 2]
> write.csv(cbind(dataset,
+     pr), file = "scores_090601.csv",
+     row.names = FALSE)
> head(cbind(Actual = dataset$TARGET_Adjusted,
+     Predicted = pr))

  Actual Predicted
1      0     0.022
2      0     0.034
3      0     0.002
4      1     0.802
5      1     0.782
6      0     0.158
```

As an aside, we can see the random forest model is doing okay on these few observations.

In practise (e.g., in the Australian Taxation Office) once model deployment has been approved the model is deployed into a secure environment. It is scheduled regularly to be applied to new data using a script that is very similar to that above (using the **littler** package for GNU/Linux). The data is obtained from a data warehouse and the results populate a data warehouse table which is then used to automatically generate work items for auditors to action.

Export to PMML

An alternative approach to deployment is to export the model so that it can be imported into other software for prediction on new data.

We have experimented with exporting random forests to C++ code. This has been demonstrated running over millions of rows of new data in a data warehouse in seconds.

Exporting to a variety of different languages, such as C++, is not an efficient approach to exporting models. Instead, exporting to a standard representation, which other software can also export, makes more sense. This standard representation can then be exported to a variety of other languages.

The Predictive Model Markup Language (Data Mining Group, 2008) provides such a standard language for representing data mining models. PMML is an XML based standard that is supported, to some extent, by the major commercial data mining vendors and many open source data mining tools.

The **pmml** package for R was separated from the **rattle** package to allow its independent development with contributions from a broader community. PMML models generated by Rattle, using the **pmml** package, can be imported into a number of other products, including Teradata Warehouse Miner (which converts models to SQL for execution), Information Builders' WebFocus (which converts models to C code for execution on over 30 platforms), and Zementis' ADAPA tool for online execution.

The **Export** button (whilst displaying a model within the **Model** tab) will export a model as PMML.

Log

A GUI is not as complete and flexible as a full programming language. Rattle is sufficient for many data miners, providing a basic point-and-click environment for quick and consistent data mining, gaining much from the breadth and depth of R. However, a professional data miner will soon find the need to go beyond the assumptions embodied in Rattle. Rattle supports this through the **Log** tab.

As mentioned above, a log of the R commands that Rattle constructs are exposed through the **Log** tab. The intention is that the R commands be available for copying into the R console so that where Rattle only exposes a limited number of options, further options can be tuned via the R console.

The **Log** tab captures the commands for later execution and is also educational. Informative comments are included to describe the steps involved. The intention is that it provide a tutorial introduction to using R from the command line, where we obtain a lot more power.

The text that appears at the top of the **Log** tab is shown in Figure 12. Commentary text is preceded with R's comment character (the #), with R commands in between.

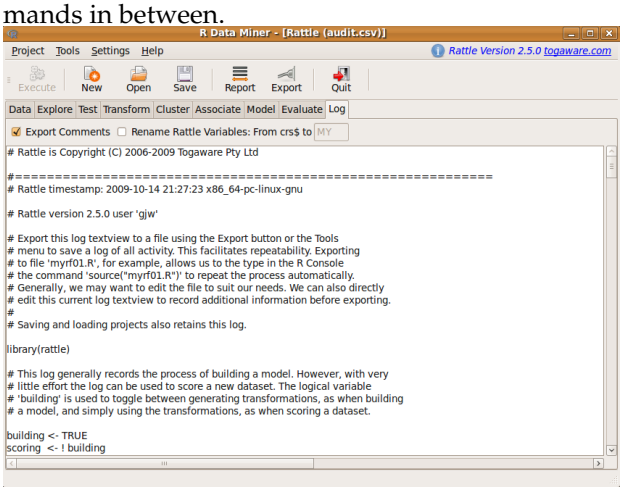

Figure 12: Rattle **Log**.

The whole log can be exported to a script file (with a '.R' filename extension) and then loaded into R or an R script editor (like Emacs/ESS or Tinn-R) to repeat the exact steps of the Rattle interactions. In general, we will want to review the code and fine-tune it to suit our purposes. After exporting the **Log** tab into a file, with a filename like 'myrf01.R', we can have the file execute as a script in R with:

```
> source("myrf01.R")
```

Help

The **Help** menu provides access to brief descriptions of the functionality of Rattle, structured to reflect the user interface. Many of the help messages then provide direct access to the underlying documentation for the various packages involved.

Future

Rattle continues to undergo development, extending in directions dictated by its actual use in data mining projects, and from suggestions and code offered by its user population. Here we mention some of the

experimental developments that may appear in Rattle over time.

A number of newer R packages provide capabilities that can enhance Rattle significantly. The **party** package and associated efforts to unify the representation of decision tree models across R is an exciting development. The **caret** package offers a unified interface to running a multitude of model builders, and significant support for tuning models over various parameter settings. This latter capability is something that has been experimented with in Rattle, but not yet well developed.

A text mining capability is in the pipeline. Current versions of Rattle can load a corpus of documents, transform them into feature space, and then have available all of Rattle's capabilities. The loading of a corpus and its transformation into feature space relies on the **tm** package (Feinerer, 2008).

Time series analysis is not directly supported in Rattle. Such a capability will incorporate the ability to analyse web log histories and observations of many entities over time.

Spatial data analysis is another area of considerable interest, often at the pre-processing stage of data mining. The extensive work completed for spatial data analysis with R (Bivand et al., 2008) may provide the basis for extending Rattle in this direction.

Further focus on missing value imputation is likely, with the introduction of more sound approaches, including k-nearest neighbours and multiple imputation.

Initial support for automated report generation using the **odfWeave** package is included in Rattle (through the Report button). Standard report templates are under development for each of the tabs. For the **Data** tab, for example, the report provides plots and tables showing distributions and basic statistics.

The Rattle code will remain open source and others are welcome to contribute. The source code is hosted by Google Code (`http://code.google.com/p/rattle/`). The Rattle Users mailing list (`http://groups.google.com/group/rattle-users`) is also hosted by Google. An open source reference book is also available (Williams, 2009a).

Acknowledgements

A desire to see R in the hands of many more of my colleagues at the Australian Taxation Office lead to the development of Rattle. I thank Stuart Hamilton, Frank Lu and Anthony Nolan for their ongoing encouragement and feedback. Michael Lawrence's work on the **RGtk2** package provided a familiar platform for the development of GUI applications in R.

The power of Rattle relies on the contributions of the open source community and the underlying R packages. For the online version of this article, follow the package links to find many of those who deserve very much credit.

We are simply standing on the shoulders of those who have gone before us, potentially providing new foundations for those who follow this way.

Bibliography

F. Andrews. *latticist: A graphical user interface for exploratory visualisation*, 2008. URL `http://latticist.googlecode.com/`. R package version 0.9-42.

R. S. Bivand, E. J. Pebesma, and V. Gómez-Rubio. *Applied Spatial Data Analysis with R*. Use R! Springer, New York, 2008. ISBN 978-0-387-78170-9.

L. Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

S. van Buuren and K. Groothuis-Oudshoorn. MICE: Multivariate Imputation by Chained Equations in R *Journal of Statistical Software*, forthcoming, 2009. URL `http://CRAN.R-project.org/package=mice`.

X. Chen, G. Williams, and X. Xu. Open source data mining systems. In Z. Huang and Y. Ye, editors, *Proceedings of the Industry Stream, 11th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD07)*, 2007.

W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, New Jersey, 1993. ISBN 0-9634884-0-6.

D. Cook and D. F. Swayne. *Interactive and Dynamic Graphics for Data Analysis*. Springer, 2007.

Data Mining Group. PMML version 3.2. WWW, 2008. URL `http://www.dmg.org/pmml-v3-2.html`.

I. Feinerer. An introduction to text mining in R. *R News*, 8(2):19–22, Oct. 2008.

A. Guazzelli, M. Zeller, W.-C. Lin, and G. Williams. Pmml: An open standard for sharing models. *The R Journal*, 1(1):60–65, May 2009. URL `http://journal.r-project.org/2009-1/RJournal_2009-1_Guazzelli+et+al.pdf`.

M. Hahsler, B. Gruen and K. Hornik arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14/15, 2005.

F. E. Harrell Jr and with contributions from many other users. *Hmisc: Harrell Miscellaneous*, 2009 URL `http://CRAN.R-project.org/package=Hmisc`. R package version 3.7-0.

J. Honaker, G. King, and M. Blackwell. *Amelia: Amelia II: A Program for Missing Data*, 2009. URL `http://CRAN.R-project.org/package=Amelia`. R package version 1.2-13.

K. Hornik, C. Buchta, and A. Zeileis. Open-Source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009.

T. Hothorn, K. Hornik and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

M. Lawrence and D. T. Lang. Rgtk2—a GUI toolkit for R. *Statistical Computing and Graphics*, 17(1), 2006.

F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. URL http://www.stat.uni-muenchen.de/~leisch/Sweave. ISBN 3-7908-1517-9.

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3), 2002.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Use R! Springer, New York, 2008. ISBN 978-0-387-75968-5.

T. Sing, O. Sander, N. Beerenwinkel and T. Lengauer. *ROCR: Visualizing the performance of scoring classifiers*, 2009. URL http://CRAN.R-project.org/package=ROCR. R package version 1.0-3.

T. M. Therneau and B. Atkinson. R port by B. Ripley. *rpart: Recursive Partitioning*, 2009. URL http://CRAN.R-project.org/package=rpart. R package version 3.1-45.

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12), 2007.

H. Wickham. *ggplot2 Elegant Graphics for Data Analysis*. Use R! Springer, New York, 2008. ISBN 978-0-387-75968-5.

H. Wickham, M. Lawrence, D. T. Lang, and D. F. Swayne. An introduction to rggobi. *R News*, 8(2): 3–7, Oct. 2008.

G. J. Williams. *The Data Mining Desktop Survival Guide*. Togaware, 2009a. URL http://datamining.togaware.com/survival.

G. J. Williams. *rattle: A graphical user interface for data mining in R*, 2009b. URL http://cran.r-project.org/package=rattle. R package version 2.4.55.

G. J. Williams. Combining decision trees: Initial results from the MIL algorithm. In J. S. Gero and R. B. Stanton, editors, *Artificial Intelligence Developments and Applications*, pages 273–289. Elsevier Science Publishers B.V. (North-Holland), 1988.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005. URL http://www.cs.waikato.ac.nz/~ml/weka/book.html.

D. Wuertz and many others. *fBasics: Rmetrics - Markets and Basic Statistics*, 2009 URL http://CRAN.R-project.org/package=fBasics. R package version 2100.78.

Graham Williams
Togaware Pty Ltd
Graham.Williams@togaware.com

sos: Searching Help Pages of R Packages

by Spencer Graves, Sundar Dorai-Raj and Romain François

Abstract: The **sos** package provides a means to quickly and flexibly search the help pages of contributed packages, finding functions and datasets in seconds or minutes that could not be found in hours or days by any other means we know. Its `findFn` function accesses Jonathan Baron's *R Site Search* database and returns the matches in a data frame of class `"findFn"`, which can be further manipulated by other **sos** functions to produce, for example, an Excel file that starts with a summary sheet that makes it relatively easy to prioritize alternative packages for further study. As such, it provides a very powerful way to do a literature search for functions and packages relevant to a particular topic of interest and could become virtually mandatory for authors of new packages or papers in publications such as *The R Journal* and the *Journal of Statistical Software*.

Introduction

The **sos** package provides a means to quickly and flexibly search the help pages of contributed packages, finding functions and datasets in seconds or minutes that could not be found in hours or days by any other means we know.

The main capability of this package is the `findFn` function, which scans the "function" entries in Jonathan Baron's *R site search* database (Baron, 2009) and returns the matches in a data frame of class `"findFn"`. Baron's site is one of five search capabilities currently identified under "Search" from the main `http://www.r-project.org/` web site. It includes options to search the help pages of R packages contributed to CRAN (the Comprehensive R Archive Network) plus a few other publicly available packages, as well as selected mailing list archives—primarily R-help. The `findFn` function focuses only on the help pages in this database. (CRAN grew from 1700 contributed packages and bundles on 2009-03-11 to 1954 on 2009-09-18, adding over 40 packages per day, an annual growth rate of 31 percent.)

The `print` method for objects of class `"findFn"` displays the results as a table in a web browser with links to the individual help pages, sorted by package, displaying the results from the package with the most matches first. This behaviour differs from that of the `RSiteSearch` function in the **utils** package in more ways than the sort order. First, `findFn` returns the results in R as a data frame, which can be further manipulated. Second, the ultimate display in a web browser is a table, unlike the list produced by `RSiteSearch`.

Other **sos** functions provide summaries with one line for each package, support the union and intersection of `"findFn"` objects, and translate a `"findFn"` object into an Excel file with three sheets: (1) **PackageSum2**, which provides an enhanced summary of the packages with matches, (2) the **findFn** table itself, and (3) the **call** used to produce it.

Three examples are considered below: First we find a data set containing a variable `Petal.Length`. Second, we study R capabilities for splines, including looking for a function named `spline`. Third, we search for contributed R packages with capabilities for solving differential equations.

Finding a variable in a data set

Chambers (2009, pp. 282-283) uses a variable `Petal.Length` from a famous Fisher data set but without naming the data set nor indicating where it can be found nor even if it exists in any contributed R package. The sample code he provides does not work by itself. To get his code to work to produce his Figure 7.2, we must first obtain a copy of this famous data set in a format compatible with his code.

To look for this data set, one might first try the `help.search` function. Unfortunately, this function returns nothing in this case:

```
> help.search('Petal.Length')
No help files found ...
```

When this failed, many users might then try `RSiteSearch('Petal.Length')`. This produced 80 matches when it was tried one day (and 62 matches a few months later). `RSiteSearch('Petal.Length', 'function')` will identify only the help pages. We can get something similar and for many purposes more useful, as follows:

```
> library(sos)
> PL <- findFn('Petal.Length')
```

`PL` is a data frame of class `"findFn"` identifying all the help pages in Jonathan Baron's data base matching the search term. An alias for `findFn` is `???`, and this same search can be performed as follows:

```
> PL <- ???Petal.Length
```

(The `???` alias only works in an assignment, so to print immediately, you need something like `(PL <- ???Petal.Length)`.)

This data frame has columns `Count`, `MaxScore`, `TotalScore`, `Package`, `Function`, `Date`, `Score`, `Description`, and `Link`. `Function` is the name of the help page, not the name of the function, as multiple functions may be documented on a single help page, and some help pages document other things such as data sets. `Score` is the index of the strength of the

match. It is used by Baron's search engine to decide which items to display first. `Package` is the name of the package containing `Function`. `Count` gives the total number of matches in `Package` found in this `findFn` call. By default, the `findFn` object is sorted by `Count`, `MaxScore`, `TotalScore`, and `Package` (to place the most important `Package` first), then by `Score` and `Function`.

The `summary` method for such an object prints a table giving for each `Package` the `Count` (number of matches), `MaxScore` (max of `Score`), `TotalScore` (sum of `Score`), and `Date`, sorted like a Pareto chart to place the `Package` with the most help pages first:

```
> # the following table has been
> # manually edited for clarity
> summary(PL)

Total number of matches: 27
Downloaded 27 links in 14 packages.
Packages with at least 1 match using search
                     pattern 'Petal.Length':
Package Count MaxScore TotalScore       Date
yaImpute    8        1          8 2009-08-16
<...>
datasets    1        2          2 2009-07-09
<...>
```

(The `Date` here is the date that this package was added to Baron's database.)

One of the listed packages is **datasets**. Since it is part of the default R distribution, we decide to look there first. We can select that row of PL just like we would select a row from any other data frame:

```
> PL[PL$Package == 'datasets', 'Function']
[1] iris
```

Problem solved in less than a minute! Any other method known to the present authors would have taken substantially more time.

Finding packages with spline capabilities

Almost four years ago, the lead author of this article decided he needed to learn more about splines. A literature search began as follows:

```
> RSiteSearch('spline')
```

(using the `RSiteSearch` function in the **utils** package). While preparing this manuscript, this command identified 1526 documents one day. That is too many. It can be restricted to functions as follows:

```
> RSiteSearch('spline', 'fun')
```

This identified only 739 one day (631 a few months earlier). That's an improvement over 1526 but is still too many. To get a quick overview of these 739, we can proceed as follows:

```
> splinePacs <- findFn('spline')
```

This downloaded a summary of the 400 highest-scoring help pages in the 'RSiteSearch' data base in roughly 5-15 seconds, depending on the speed of the Internet connection. To get all 739 matches, increase the `maxPages` argument from its default 20:

```
> splineAll <- findFn('spline',
+     maxPages = 999)
```

If we want to find a function named `spline`, we can proceed as follows:

```
> selSpl <- (splineAll[, 'Function'] ==
+    'spline')
> splineAll[selSpl, ]
```

This has 0 rows, because there is no help page named `spline`. This does not mean that no function with that exact name exists, only that no help page has that name.

To look for help pages whose name includes the characters '`spline`', we can use `grepFn`:

```
> grepFn('spline', splineAll,
+     ignore.case = TRUE)
```

This returned a `"findFn"` object identifying 78 help pages. When this was run while preparing this manuscript, the sixth row was `lspline` in the **assist** package, which has a `Score` of 1. (On another day, the results could be different, because CRAN changes over time.) This was the sixth row in this table, because it is in the **assist** package, which had a total of 34 help pages matching the search term, but this was the only one whose name matched the `pattern` passed to `grepFn`.

We could next `print` the `splineAll` `"findFn"` object. However, it may not be easy to digest a table with 739 rows.

`summary(splineAll)` would tell us that the 739 help pages came from 191 different packages and display the first `minPackages = 12` such packages. (If other packages had the same number of matches as the twelfth package, they would also appear in this summary.)

A more complete view can be obtained in MS Excel format using the `writeFindFn2xls` function:

```
> writeFindFn2xls(splineAll)
```

(`findFn2xls` is an alias for `writeFindFn2xls`. We use the longer version here, as it may be easier to remember.)

If either the **WriteXLS** package and compatible Perl code are properly installed or if you are running Windows with the **RODBC** package, this produces an Excel file in the working directory named 'splineAll.xls', containing the following three worksheets:

- The **PackageSum2** sheet includes one line for each package with a matching help page, enhanced by providing information for locally installed packages not available in the "findFn" object.
- The **findFn** sheet contains the search results.
- The **call** sheet gives the call to findFn that generated these search results.

If writeFindFn2xls cannot produce an Excel file with your installation, it will write three 'csv' files with names 'splineAll-sum.csv', 'splineAll.csv', and 'splineAll-call.csv', corresponding to the three worksheets described above. (Users who do not have MS Excel may like to know that Open Office Calc can open a standard 'xls' file and can similarly create such files (Openoffice.org, 2009).)

The **PackageSum2** sheet is created by the PackageSum2 function, which adds information from installed packages not obtained by findFn. The extended summary includes the package title and date, plus the names of the author and the maintainer, the number of help pages in the package, and the name(s) of any vignettes. This can be quite valuable in prioritizing packages for further study. Other things being equal, we think most people would rather learn how to use a package being actively maintained than one that has not changed in five years. Similarly, we might prefer to study a capability in a larger package than a smaller one, because the rest of the package might provide other useful tools or a broader context for understanding the capability of interest.

These extra fields, package title, etc., are blank for packages in the "findFn" object not installed locally. For installed packages, the Date refers to the *packaged* date rather than the date the package was added to Baron's database.

Therefore, the value of PackageSum2 can be increased by running install.packages (from the **utils** package) to install packages not currently available locally and update.packages() to ensure the local availability of the latest versions for all installed packages.

To make it easier to add desired packages, the **sos** package includes an installPackages function, which checks all the packages in a "findFn" object for which the number of matches exceeds a second argument minCount and installs any of those not already available locally; the default minCount is the square root of the largest Count. Therefore, the results from PackageSum2 and the **PackageSum2** sheet created by writeFindFn2xls will typically contain more information after running installPackages than before.

To summarize, three lines of code gave us a very powerful summary of spline capabilities in contributed R packages:

```
splineAll <- findFn('spline', maxPages = 999)
installPackages(splineAll)
writeFindFn2xls(splineAll)
```

The resulting 'splineAll.xls' file can help establish priorities for further study of the different packages and functions. An analysis of this nature almost four years ago led the lead author to the **fda** package and its companion books, which further led to a collaboration that has produced joint presentations at three different conferences and a joint book (Ramsay et al., 2009).

Combining search results

The lead author of this article recently gave an invited presentation on "Fitting Nonlinear Differential Equations to Data in R"[1]. A key part of preparing for that presentation was a search of contributed R code, which proceeded roughly as follows:

```
> de <- findFn('differential equation')
> des <- findFn('differential equations')
> de. <- de | des
```

The object de has 53 rows, while des has 105. If this search engine were simply searching for character strings, de would be larger than des, rather than the other way around. The last object de. is the union of de and des; '|' is an alias for unionFindFn. The de. object has 124 rows, which suggests that the corresponding intersection must have (53+105-124) = 34. This can be confirmed via nrow(de & des). ('&' is an alias for intersectFindFn.)

To make everthing in de. locally available, we can use installPackages(de., minCount = 1). This installed all referenced packages except rmutil and a dependency Biobase, which were not available on CRAN but are included in Jonathan Baron's R Site Search data base.

Next, writeFindFn2xls(de.) produced a file 'de..xls' in the working directory. (The working directory can be identified via getwd().)

The **PackageSum2** sheet of that Excel file provided a quick summary of packages with matches, sorted to put the package with the most matches first. In this case, this first package was **deSolve**, which provides, "General solvers for initial value problems of ordinary differential equations (ODE), partial differential equations (PDE) and differential algebraic equations (DAE)". This is clearly quite relevant to the subject. The second package was **PKfit**, which is "A Data Analysis Tool for Pharmacokinetics". This may

[1] Workshop on Statistical Methods for Dynamic System Models, Vancouver, 2009: http://stat.sfu.ca/~dac5/workshop09/Spencer_Graves.html

be too specialized for general use. I therefore would not want to study this first unless my primary interest here was in pharmacokinetic models.

By studying the summary page in this way, I was able to decide relatively quickly which packages I should consider first. In making this decision, I gave more weight to packages with one or more vignettes and less weight to those where the `Date` was old, indicating that the code was not being actively maintained and updated. I also checked the conference information to make sure I did not embarrass myself by overlooking a package authored or maintained by another invited speaker.

Discussion

We have found `findFn` in the **sos** package to be very quick, efficient and effective for finding things in contributed packages. The `grepFn` function helps quickly look for functions (or help pages) with particular names. The capabilities in `unionFindFn` and `intersectFindFn` (especially via their '`|`' and '`&`' aliases) can be quite useful where a single search term seems inadequate; they make it easy to combine multiple searches to produce something closer to what is desired. An example of this was provided with searching for both "differential equation" and "differential equations".

The **PackageSum2** sheet of an Excel file produced by `writeFindFn2xls` (after also running the `installPackages` function) is quite valuable for understanding the general capabilities available for a particular topic. This could be of great value for authors to find what is already available so they don't duplicate something that already exists and so their new contributions appropriately consider the contents of other packages.

The `findFn` capability can also reduce the risk of "the researcher's nightmare" of being told after substantial work that someone else has already done it.

Users of **sos** may also wish to consult Crantastic (`http://www.crantastic.org/`), which allows users to tag, rate, and view packages. (The coverage of Crantastic includes current and orphaned CRAN packages, while Baron (2009) also includes 'most of the default packages from Bioconductor and all of Jim Lindsey's packages.')

Acknowledgments

The capabilities described here extend the power of the R Site Search search engine maintained by Jonathan Baron. Without Prof. Baron's support, it would not have been feasible to develop the features described here. Duncan Murdoch, Marc Schwarz, Dirk Eddelbuettel, Gabor Grothendiek and anonymous referees contributed suggestions for improvement, but of course can not be blamed for any deficiencies. The collaboration required to produce the current **sos** package was greatly facilitated by R-Forge (R-Forge Team, 2009). The **sos** package is part of the R Site Search project hosted there. This project also includes code for a Firefox extension to simplify the process of finding information about R from within Firefox. This Firefox extension is still being developed with the current version downloadable from `http://addictedtor.free.fr/rsitesearch`.

Bibliography

J. Baron. R site search. URL `http://finzi.psych.upenn.edu/search.html`, September 2009.

J. Chambers. *Software for Data Analysis: Programming with R*. Springer, New York, 2009.

Openoffice.org. *Open Office Calc*. Sun Microsystems, California, USA, 2009. URL `http://www.openoffice.org`.

J. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer, New York, 2009.

S. Theußl and A. Zeileis "Collaborative software development using R-Forge". *The R Journal*. 1(1):9–14.

Spencer Graves
Structure Inspection and Monitoring
San Jose, CA
`spencer.graves@prodsyse.com`

Sundar Dorai-Raj
Google
Mountain View, CA
`sdorairaj@google.com`

Romain François
Independent R Consultant
Montpellier, France
`francoisromain@free.fr`

The New R Help System

by Duncan Murdoch and Simon Urbanek

Abstract: Version 2.10.0 of R includes new code for processing '.Rd' help files. There are some changes to what is allowed, and some new capabilities and opportunities.

One of the reasons for the success of R is that package authors are required to write documentation in a structured format for all of the functions and datasets that they make visible in their packages. This means users can count on asking for help on a topic `"foo"` using either the function call `help("foo")`, or one of the shortcuts: entering `?foo` in the console, or finding help through the menu system in one of the graphical user interfaces. The quality of the help is improved by the structured format: the quality assurance tools such as `R CMD check` can report inconsistencies between the documentation and the code, undocumented objects, and other errors, and it is possible to build indices automatically and do other computations on the text.

The original help system was motivated by the help system for S (Becker et al., 1988), and the look of the input files was loosely based on LaTeX (Lamport, 1986). Perl (Wall et al., 2000) scripts transformed the input files into formatted text to display in the R console, LaTeX input files to be processed into Postscript or PDF documents, and HTML files to be viewed in a web browser.

These Perl scripts were hard to maintain, and inconsistencies crept into the system: different output formats could inadvertently contain different content. Moreover, since the scripts could only look for fairly simple patterns, the quality control software had trouble detecting many errors which would slip through and result in rendering errors in the help pages, unknown to the author.

At the useR! 2008 meeting in Dortmund, one of us (Murdoch) was convinced to write a full-fledged parser for '.Rd' files. This made it into the 2.9.0 release in April, 2009, but only as part of the quality control system: and many package authors started receiving warning and error messages. Over the summer since then several members of the R Core team (including especially Brian Ripley and Kurt Hornik, as well as the authors of this article) have refined that parser, and written renderers to replace the Perl scripts, so that now all help processing is done in R code. We have also added an HTTP server to R to construct and deliver the help pages to a web browser on demand, rather than relying on static copies of the pages.

This article describes the components of the new help system.

The Parser

In the new help system, the transformation from an '.Rd' file to a '.tex', '.html' or '.txt' file is a two-step process. The `parse_Rd()` function in the **tools** package converts the '.Rd' file to an R object with class `"Rd"` representing its structure, and the renderers convert that to the target form.

Rd Syntax

The syntax of '.Rd' files handled by the parser is in some ways more complicated than the syntax of R itself. It contains LaTeX-like markup, R-like sections of code, and occasionally code from other languages as well. Figure 1 shows a near-minimal example.

```
% Comments in .Rd files start with percent signs
\name{foo}
\alias{footopic}
\title{Title to Display at the Top of the Page}
\description{
  A short description of what is being documented.
}
\usage{
  foo(arg = "\n")
}
\arguments{
  \item{arg}{the first argument.}
}
\seealso{
  \code{\link{bar}}.
}
\examples{
  ## call foo then \link{bar} in a loop
  for (i in 1:10) {
    foo(1)
    bar(2)
  }
}
\keyword{example}
```

Figure 1: The contents of a simplified '.Rd' file.

Like LaTeX, a comment is introduced by a percent sign (`%`), and markup is prefixed with a backslash (`\`). Braces (`{}`) are used to delimit the arguments to markup macros. However, notice that in the `\examples{}` section, we follow R syntax: braces are used to mark the body of the `for` loop.

In fact, there are three different modes that the parser uses when parsing the '.Rd' file. It starts out in a LaTeX-like mode, where backslashes and braces are used to indicate macros and their arguments. Some macros (e.g. `\usage{}` or `\examples{}`) switch into R-like mode: macros are still recognized, but braces are used in the code, and don't necessarily indicate the arguments to macros. The third mode is mostly verbatim: it doesn't recognize any macros at

all. It is used in a few macros like `\alias{}`, where we might want to set an alias involving backslashes, for instance. There are further complications in that strings in quotes in R-like mode (e.g. `"\n"`) are handled slightly differently again, so the newline is not treated as a macro, and R comments in R-like mode have their own special rules.

A good question to ask is why the syntax is so complicated. Largely this is a legacy of the earlier versions of the help system. Because there was no parsing, just a single-step transformation to the output format, special characters were handled on a case-by-case basis, and not always consistently. When writing the new parser, we wanted to minimize the number of previously correct '.Rd' files which triggered errors or were rendered incorrectly. The three-mode syntax described above is the result.

The syntax rules are fully described in Murdoch (2009), but most users should not need to know all the details. In almost all cases, the syntax is designed so that typing what you mean will give the result you want. A quick summary is as follows:

- The characters `\`, `%`, `{` and `}` have special meaning almost everywhere in an '.Rd' file.
- The backslash `\` is used both to introduce macros and to escape the special meaning of the other characters.
- Unless escaped, the percent `%` character starts a comment. The comment is included in the parsed object, but the renderers will not display it to the user.
- Unless escaped, the braces `{` and `}` delimit the arguments to macros. In R-like or verbatim text, they need not be escaped if they balance.
- End of line (newline) characters mark the end of pieces of text, even when following a `%` comment.
- Other whitespace (spaces, tabs, etc.) is included as part of the text, though the renderers may remove or change it.
- In R-like text, quoted strings follow R rules: the delimiters must balance, and braces within need not. Only a few macros are recognized in R strings: `\var` and those related to links. Other uses of a backslash, e.g. `"\n"` are taken to be part of the string.
- In R-like text, R comments using `#` are taken to be part of the text.
- The directives `#ifdef ... #endif` and `#ifndef ... #endif` are treated as markup, so other macros must be completely nested within them or completely contain them.

There are more than 60 macros recognized by the parser, from `\acronym` to `\verb`. Each of them takes from 0 to 3 arguments in braces, and some of them take optional arguments in brackets (`[]`). A complete list is given in Murdoch (2009), and their interpretation is described in the *Writing R Extensions* manual (R Development Core Team, 2009).

Rd Objects

The output of the `parse_Rd()` function is a list with class `"Rd"`. Currently this is mostly intended for internal use, and the only methods defined for that class are `as.character.Rd()` and `print.Rd()`. The elements of the list are the top level components of the '.Rd' file. Each element has an `"RdTag"` attribute labelling it as one of the three kinds of text, or a comment, or as a macro. There is a (currently internal) function in the **tools** package to display all of the tags. For example, using the help file from Figure 1, we get the following results.

```
> library(tools)
> parsed <- parse_Rd("foo.Rd")
> tools:::RdTags(parsed)

 [1] "COMMENT"       "TEXT"
 [3] "\\name"        "TEXT"
 [5] "\\alias"       "TEXT"
 [7] "\\title"       "TEXT"
 [9] "\\description" "TEXT"
[11] "\\usage"       "TEXT"
[13] "\\arguments"   "TEXT"
[15] "\\seealso"     "TEXT"
[17] "\\examples"    "TEXT"
[19] "\\keyword"     "TEXT"
```

The `TEXT` components between each section are simply the newlines that separate them, and can be ignored. The components labelled with macro names are themselves lists, with the same structure. For example, component 15 is the `\seealso` section, and it has this structure

```
> tools:::RdTags(parsed[[15]])

[1] "TEXT"   "TEXT"   "\\code" "TEXT"
```

and the `\code` macro within it is a list, etc. Most users will have no need to look at `"Rd"` objects at this level, but this is crucial for the internal quality assurance code, and for the renderers described in the next section. As R 2.10.0 is the first release where these objects are being fully used, there have been a number of changes since they were introduced in R 2.9.0, and there may still be further changes in future releases: so users of the internal structure should pay close attention to changes taking place on the R development trunk.

Incompatibilities

One of the design goals in writing the new parser was to accept valid '.Rd' files from previous versions. This was mostly achieved, but there are some incompatibilities arising from the new syntax rules given above. A full description is included in Murdoch (2009); here we point out some highlights.

In previous versions, the `\code{}` macro was used for both R code and code in other languages. However, the R code needs R-like parsing rules, and other languages often need verbatim parsing. Since R code is more commonly used, it was decided to restrict `\code{}` to handle R code and introduce `\verb{}` for the rest. At present, this mainly affects text containing quote marks, which must balance in a `\code{}` block as in R code. At some point in the future legal R syntax might be more strictly enforced.

The handling of `#ifdef/#ifndef` and the rules for escaping characters are now more regular.

Error Handling

One final feature of `parse_Rd()` is worth mentioning. We make a great effort to report errors in '.Rd' files at the point they occur, and because we now fully parse the file, we have detected a large number of errors that previously went unnoticed. In most cases these errors resulted in help pages that were not being displayed as the author intended, but we don't want to suddenly make hundreds of packages on CRAN unusable. The compromise we reached is as follows: when the parser detects an error in a '.Rd' file, it reports an error or warning, and attempts to recover, possibly skipping some text. The package installation code will report these messages, but will not abort an installation because of them. Authors should not ignore these messages: in most cases, they indicate that something will not be displayed as intended.

The Renderers

As of version 2.10.0, R will present help in three different formats: text, HTML, and PDF manuals. Previous versions also included compiled HTML on the Windows platform; that format is no longer supported, due partly to security concerns (Microsoft Support, 2007), partly to the fact that it will not support dynamic help, and partly to reduce the support load on the R Core team. The displayed type is now controlled by a single option rather than a collection of options: for example, to see HTML help, use `options(help_type="html")` rather than `options(htmlhelp=TRUE)`.

The three formats are produced by the functions `Rd2txt()`, `Rd2HTML()` and `Rd2latex()` in the **tools** package. The main purpose of these functions is to convert the parsed `"Rd"` objects to the output format; they also accept '.Rd' files as input, calling `parse_Rd()` if necessary.

There are two other related functions in **tools**. The `Rd2ex()` function extracts example code from a help page, and `checkRd()` performs checks on the contents of an `"Rd"` object. The latter checks for the presence of all necessary sections (at most once in some cases, e.g. for the `\title{}`). It is used to report errors and warnings during `R CMD check` processing of a package.

One big change from earlier versions of R is that these functions do their rendering on request. In earlier versions, all help processing was done when the package was installed, and the text files, HTML files, and PDF manuals were installed with the package. Now the `"Rd"` objects are produced at install time, but the human-readable displays are produced at the time the user asks to see them. As described below, this means that help pages can include information computed just before the page is displayed. Not much use is made of this feature in 2.10.0, but the capability is there, and we expect to make more use of it in later releases of base R, and to see it being used in user-contributed packages even sooner.

Another advantage of "just-in-time" rendering is that cross-package links between help pages are handled better. In previous versions of R, the syntax

```
\link[stats]{weighted.mean}
```

meant that the HTML renderer should link to the *file* named 'weighted.mean.html' in the **stats** package, whereas

```
\link{weighted.mean}
```

meant to link to whatever file corresponded to the *alias* `weighted.mean` in the current package. This difference was necessary because the installer needed to build links at install time, but it could not necessarily look up topics outside of the current package (the target package might not be installed yet). Unfortunately, the inconsistency was a frequent source of errors, even in the base packages. Now that calculation can be delayed until rendering time, R 2.10.0 supports both kinds of linking. For back-compatibility, it gives priority to linking by filename, but if that fails, it will try to link by topic.

The HTTP server

In order to produce the help display in a web browser when the user requests, it was necessary to add an HTTP server to R.

At a high level the goal of the server is to accept connections from browsers, convert each HTTP request on a TCP/IP port into a call to an R function and deliver the result back to the browser. Since the main use is to provide a dynamic help system based

on the current state of the R session (such as packages loaded, methods defined, ...), it is important that the function is evaluated in the current R session. This makes the implementation more tricky as we could not use regular socket connections and write the server entirely in R.

Instead, a general asynchronous server infrastructure had to be created which handles multiple simultaneous connections and is yet able to synchronize them into synchronous calls of the R evaluator. The processing of each connection is dispatched to an asynchronous worker which keeps track of the state of the connection, collects the entire HTTP request and issues a message to R to process the request when complete. The request is processed by calling the `httpd` function with two arguments: path from the URL and query parameters. The query parameters are parsed from the query string into a named string vector and decoded, so for example `text=foo%3f&n=10` is passed as `c(text="foo?", n="10")`.

The `httpd()` function is expected to produce output in the form of a list that is meaningful to the browser. The list's elements are interpreted in sequence as follows: *payload*, *content-type*, *headers* and *status code*. Only the payload is mandatory so the returned list can have one to four elements. The content type specifies the MIME-type of the payload (default is "text/html"), headers specify optional HTTP response headers as a named character vector (without CR/LF) and the error code is an integer specifying the HTTP status code. The payload must be either a string vector of length one or a raw vector. If the payload element is a string vector named "file" then the string is interpreted as an absolute path to a file to be sent as the payload (useful for static content). Otherwise the payload is sent to the browser in verbatim. The `httpd` function is evaluated in a `tryCatch` block such that errors are returned as a string from the function, resulting in a 500 status code (internal server error). An example of a simple `httpd` function is listed below:

```
httpd <- function(path, query, ...) {
  if (is.null(query)) query <- character(0)
  pkg <- query['package']
  if (is.na(pkg)) pkg <- 'base'
  fn <- system.file(path, package = pkg)
  if (file.exists(fn)) return(list(file = fn))
  list(paste("Cannot find", path),
       "text/html", NULL, 404L)
}
```

The signature of the `httpd` function should include ... for future extensions. The above function checks for the `package` query parameter (defaulting to "base") then attempts to find a file given by `path` in that package. If successful the content of the file is served (as default text/html), otherwise a simple error page is created with a 404 HTTP status code "not found".

Although the HTTP server is quite general and could be used for many purposes, R 2.10.0 has the limitation of one server per R instance which makes it currently dedicated to the dynamic help. The user may set `options("help.ports")` to control which IP port is used by the server. If not set, the port is assigned randomly to avoid collisions between different R instances.

New Features in '.Rd' Files

R Expressions in Help

As mentioned previously, help output is now being produced just before display, and it is possible for the author to customize the display at that time. This is supported by the new macro `\Sexpr{}`, which is modelled after the macro of the same name in current versions of Sweave (Leisch, 2002).

Unlike Sweave, in '.Rd' files the `\Sexpr{}` macro takes optional arguments to control how it is displayed, and when it is calculated. For example, `\Sexpr[results=verbatim,echo=TRUE]{x<-10;x^2}` will result in a display similar to

```
> x<-10;x^2

[1] 100
```

when the help page is displayed.

Some of the options are similar to Sweave, but not always with the same defaults. For example the options (with defaults) `eval=TRUE`, `echo=FALSE`, `keep.source=TRUE`, and `strip.white=TRUE` do more or less the same things as in Sweave. That is, they control whether the code is evaluated, echoed, reformatted, and stripped of white space before display, respectively.

There are also options that are different from Sweave. The `results` option has the following possible values:

results=text (the default) The result should be inserted into the text at the current point.

results=verbatim Print the result as if it was executed at the console.

results=hide No result should be shown.

results=rd The result should be parsed as if it was '.Rd' file code.

The `stage` option controls when the code is run. It has the following values:

stage=build The code should be run when building the package. This option is not currently implemented, but is allowed; the code will not be executed in R 2.10.0.

stage=install (the default) The code is run when installing the package from source, or when building a binary version of the package.

stage=render The code is run just before the page is rendered.

The `\Sexpr{}` macro enables a lot of improvements to the help system. It will allow help pages to include examples with results inline, so that they can be mixed with descriptions, making explanations clearer. The just-in-time rendering will also allow help pages to produce information customized to a particular session: for example, it would be helpful to link from a page about a class to methods which mention it in their signatures. It will also be possible to prototype new types of summary pages or indices for the whole help system, without requiring changes to base R.

Conditional markup

The help files have supported `#ifdef/#ifndef` conditionals for years, to allow selective inclusion of text depending on the computing platform where R is run. With version 2.10.0 more general conditional selection has been added: the `\if{}{}` and `\ifelse{}{}{}` macros. The first argument to these describes the condition, the second is code to be included at render time if that condition holds, and the third if it does not.

The most common use of the conditionals is foreseen to be to select displays depending on the output format, to extend the idea of the `\eqn` and `\deqn` macros from previous versions. To support this use, the condition is expressed as a comma-separated list of formats, chosen from `example`, `html`, `latex`, `text`, `TRUE` and `FALSE`. If the current output format or `TRUE` is in the list, the condition is satisfied. The logicals would normally be the result of evaluating an `\Sexpr{}` expression, so general run-time conditions are possible.

Verbatim output

Two new macros have been added to support output of literal text. The `\verb{}` macro parses its argument as verbatim text, and the renderers output it as parsed, inserting whatever escapes and conversions are necessary so that it renders the same in all formats. The `\out{}` macro works at a lower level. It also parses its argument as verbatim text, but the renderers output it in raw form, without any processing. It would almost certainly be used in combination with `\if` or `\ifelse`. For example, to output a Greek letter alpha, the markup

```
\ifelse{html}{\out{&alpha;}}{
\eqn{\alpha}{alpha}}
```

could be used: this will output `&alpha;` when producing HTML and `\alpha` when producing LaTeX, both of which will render as α; it will output `alpha` when producing text.

The Future

One of the development guidelines for R is that we don't introduce many new features in patch releases, so R 2.10.1 is not likely to differ much from what is described above. However, we would expect small changes to the rendering of help pages as the renderers become more polished.

For version 2.11.0 or later, we would expect some of the following to be implemented. The order of presentation is not necessarily the order in which they will appear, and some may never appear.

- The `\Sexpr{}` macro will likely develop in several ways:
  - We will likely add in the processing of `stage=build` macros when the `R CMD build` code is rewritten in R.
  - We will likely change the `prompt()`, `package.skeleton()` and related functions to make use of `\Sexpr{}` macros, e.g. to display the version of a package, or to list the classes supporting a method, etc.
  - We will make information about the run-time environment available to the `\Sexpr{}` code, so that it can tailor its behaviour to the context in which it is being run.
  - We would expect users to develop contributed packages to provide convenient functions to use in `\Sexpr{}` macros. Some of these may need changes to base R.
  - We would like to allow figures to be inserted into help displays.
- The quality control checks will continue to evolve, as we notice common errors and add code to `checkRd()` to detect them. There is already support for spell checking of '.Rd' files in the `aspell()` function.
- The '.Rd' format is very idiosyncratic, and not many tools exist outside of R for working with it. There may be an advantage to switching to a different input format (e.g. one based on XML) in order to make use of more standard tools. Such a change would require conversion of the thousands of existing help pages already in '.Rd' format; the new parser may enable automatic translation if anyone wants to explore this possibility.
- The HTTP server used in the help system is quite general, but is currently limited to serving R help pages. It may be extended to allow more general use in the future.

Changing to the new help system has already helped to diagnose hundreds of errors in help pages that slipped by in the previous version, and diagnostic messages are more informative than they were. As the `\Sexpr{}` macro is used in core R and by package writers, we will see better help than ever before, and the HTTP server will open up many other possibilities for R developers.

Bibliography

R. A. Becker, J. M. Chambers, and A. R. Wilks. *The New S Language*. Wadsworth, Pacific Grove, California, 1988.

L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1986.

F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. URL http://www.stat.uni-muenchen.de/~leisch/Sweave. ISBN 3-7908-1517-9.

Microsoft Support. Ms05-026: A vulnerability in html help could allow remote code execution, 2007. Retrieved from http://support.microsoft.com/kb/896358 on November 5, 2009.

D. Murdoch. Parsing Rd files. 2009. URL http://developer.r-project.org/parseRd.pdf.

R Development Core Team. *Writing R Extensions*, 2009. Manual included with R version 2.10.0.

L. Wall, T. Christiansen, and J. Orwant. *Programming Perl*. O'Reilly Media, third edition, 2000.

Duncan Murdoch
Dept. of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario, Canada
murdoch@stats.uwo.ca

Simon Urbanek
AT&T Labs – Statistics Research
Florham Park, NJ, USA
urbanek@research.att.com

Conference Review: DSC 2009

by Peter Dalgaard

The sixth international workshop on Directions in Statistical Computing (DSC 2009) took place at the Center for Health and Society, University of Copenhagen, Denmark, 13th–14th of July 2009.

The aim of the workshop series is to provide a platform for exchanging ideas about developments in statistical computing (rather than 'only' the usage of statistical software for applications).

We had 58 participants from many different countries. We listened to 26 talks in 11 sessions. Ten of the talks were given in plenary sessions and the rest in two parallel tracks. In keeping with the egalitarian nature of the conference, all talks had the same length, 35 minutes, including discussion.

The weather was slightly cool for high Summer in Copenhagen, but nicely dry, and the Conference dinner at the beautifully situated Furesø Marina turned out to be a very pleasant experience indeed.

Selected papers from the conference will be published as a special issue of Computational Statistics. The peer review process is currently ongoing.

The organizing committee, *Peter Dalgaard*, *Claus Ekstrøm*, *Klaus K. Holst* and *Søren Højsgaard* would like to thank

- The Programme Committee: Roger Bivand, Göran Broström, Peter Dalgaard, and Balasubramanian Narasimhan
- Our host, the Department of Biostatistics at the University of Copenhagen
- "Statistical Methods for Complex and High-dimensional Models" (a.k.a. the interfaculty Statistics Network at the University of Copenhagen)
- The R Foundation
- REvolution Computing for sponsoring the conference bags
- NSF(USA) for travel grants
- DIS Congress Service A/S

On behalf of the Organizing Committee.

Peter Dalgaard
Department of Biostatistics
Øster Farimagsgade 5, Entr. B
DK-1014 Copenhagen K
p.dalgaard@biostat.ku.dk

Conference Review: WZUR(2.0) – The Second Meeting of Polish R Users

by Przemyslaw Biecek

The second Polish R users meeting was held 3-4 October 2009 in Warsaw. This year, as well as the year before, the meeting was called WZUR. This is an acronym for "Warszawski Zlot Użytkowników R" - in rough translation "R users gathering that is held in Warsaw". The word "WZUR" has also another meaning in Polish, corresponding to English word "equation".

The mission of WZUR meetings is to gather the R-addicted (or R-using, contrary to R-abusing) individuals, not only from R community, and give them an opportunity to discuss and exchange ideas, experiences and opinions. The big success of the meeting is appearance of people from various areas of interest. This year the conference hosted talks in archeology, medicine and bioinformatics, and animal genetics. Presentations were also made by pure theoretical mathematicians, applied statisticians and computer scientists. The talks presented creative and interesting applications of R in both applied and pure sciences.

This year the conference was split in two days. The first day of the conference was reserved for two half-day workshops focused on Gaussian and generalized mixed models and their relation to penalized regression. We have the great pleasure of attending two lectures about theory behind mixed modeling and lectures which presented various applications of Gaussian and generalized mixed modeling with real data examples.

The second day consisted of a series of shorter talks, distributed into 4 sessions, namely "R Kaleidoscope" , "Applications in analyses of real data", "Mathematical Statistics" and "How to do it in R". The session "How to do it in R" was a new, but successful idea. During the session users briefly presented their field of interest, problems they bump into and how they had successfully faced the challenges using R. These problems mostly concerned the R environment or were related to statistical foundation of some procedures or more complicated types of analyses. Each presenter has been generously flooded with useful hints (one at least).

The conference was conducted in Polish but all materials are available in English. They are published on the website http://www.biecek.pl/WZUR/indexeng.html (English version) and http://www.biecek.pl/WZUR/ (Polish version). Almost all talks and workshops were recorded and the video files are freely available on the above named site. Be aware that these video files are quite large, so expect on average 100MB per talk.

We would like to thank our generous sponsors and supporters: the Netezza company and the Warsaw University for their generous financial support.

We are looking forward to the next WZUR and always warmly welcome new participants (and sponsors!).

Przemyslaw Biecek
Institute of Applied Mathematics
Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
Banacha 2, 02-097 Warszawa, Poland
przemyslaw.biecek@gmail.com

R Changes: 2.9.1-2.10.0 Patched

by the R Core team

Changes in R 2.10 patched

New features

- The PCRE library has been updated to version 8.00.
- 'R CMD INSTALL' has new options '--no-R', '--no-libs', '--no-data', '--no-help', '--no-demo', '--no-exec', and '--no-inst' to suppress installation of the specified part of the package. These are intended for special purposes (e.g. building a database of help pages without fully installing all packages).
- The documented line-length limit of 4095 bytes when reading from the console now also applies also to `parse(file="")` (which previously had a limit of around 1024 bytes).
- A Bioconductor mirror can be set for use by `setRepositories()` via the option `"BioC_mirror"`, e.g. the European mirror can be selected by `options(BioC_mirror="http://bioconductor.statistik.tu-dortmund.de")`
- Double-clicking in a `tk_select.list()` list box now selects the item and closes the list box (as happens on the Windows `select.list()` widget).

Installation changes

- configure will be able to find a usable libtiff in some rare circumstances where it did not previously (where libtiff needed to be linked explicitly against -ljpeg).
- Making 'refman.pdf' works around a problem with the indexing with hyperref 6.79d and later.

Deprecated & defunct

- The `extended` argument is deprecated in `strsplit()`, `grep()`, `grepl()`, `sub()`, `gsub()`, `regexpr()` and `gregexpr()` (not just the value `extended = FALSE`) and will be removed in R 2.11.0.

Bug fixes

- `trigamma(x)` and other `psigamma(x, n)` calls are now accurate for very large `abs(x)`. (PR#14020)
- `[g]sub(perl=FALSE, fixed=FALSE)` could use excessive stack space when used with a very long vector containing some non-ASCII strings.
- The default method of `weighted.mean(na.rm = TRUE)` did not omit weights for `NA` observations in 2.10.0. (PR#14032)
- `[g]regexpr(pattern, fixed = TRUE)` returned match positions in bytes (not characters) in an MBCS locale if `pattern` was a single byte.

  `[g]sub(fixed = TRUE)` with a single-byte pattern could conceivably have matched part of a multibyte character in a non-UTF-8 MBCS.
- `findLineNum()` and `setBreakpoint()` would sometimes fail if the specified file was not in the current directory.
- Package tcltk's `demo(tkdensity)` was broken in 2.9.0 when `demo()` was changed to set `par(ask = TRUE)`.
- `gsub()` with backrefs could fail on extremely long strings (hundreds of thousands of characters) due to integer overflow in a length calculation.
- `abline(*, untf=TRUE)` now uses a better x-grid in log-scale, e.g., for `plot(c(1,300), c(1,300), log="xy")` ; `abline(4,1, untf=TRUE)`.
- `detach/unloadNamespace()` arrange to flush the package's lazyload cache of R objects once the package/namespace is no longer needed.
- There have been small fixes to the rendering of help, e.g. `\command` is now rendered verbatim (so e.g. '--' is not interpreted, PR#14045).

  Also, there are many small changes to help files where the new converters were not rendering them in the same way as before.
- `available.packages()` would fail when run on a repository with no packages meeting the filtering conditions. (PR#14042)
- `rep(x, times, each = 2)` gave invalid results when the `times` argument is a vector longer than 'x'. Reported by Bill Dunlap.
- An error when `unloadNamespace()` attempted to run the `.onUnload()` function gave an error in the reporting function and so was not reported properly.
- Text help rendering did not handle very long input lines properly.

- promptMethods() generated signature documentation improperly.

- pgamma(x, a, lower.tail=FALSE) and qgamma(..) are now considerably more accurate in some regions for very small 'a'. qgamma() now correctly returns 0 instead of NaN in similar extreme cases, and qgamma() no longer warns in the case of small 'a', see (PR#12324).

- unname() now also removes names from a zero length vector.

- Printing results from ls.str() no longer evaluates unevaluated calls.

- complete.cases() failed on a 0-column data frame argument. (Underlies PR#14066.)

  It could return nonsensical results if no input determined the number of cases (seen in the no-segfault tests).

- An error in nls() with a long formula could cause a segfault. (PR#14059)

- qchisq(p, df, ncp, lower.tail = FALSE) with ncp >= 80 was inaccurate for small p (as the help page said): it is now less inaccurate. (In part, PR#13999.)

  For ncp less than but close to 80, pchisq() and qchisq() are more accurate for probablilities very close to 1 (a series expansion was truncated slightly too early).

  pchisq(x, df, ncp) can no longer return values just larger than one for large values of ncp.

- intToUtf8() could fail when asked to produce 10Mb or more strings, something it was never intended to do: unfortunately Windows crashed R (other OSes reported a lack of resources). (PR#14068)

- chisq.test() could fail when given argument 'x' or 'y' which deparsed to more than one line. (Reported by Laurent Gauthier.)

- S4 methods are uncached whenever the name space containing them is unloaded (by unloadNamespace() as well as by detach(unload = TRUE)).

- The internal record-keeping by dyn.load /dyn.unload was incomplete, which could crash R if a DLL that registered .External routines had earlier been unloaded.

- bessel[JY](x, nu) with nu a negative integer (a singular case) is now correct, analogously to besselI(), see PR#13556, below.

- tools::file_path_as_absolute() doubled the file separator when applied to a file such as '/vmunix' or (on Windows) 'd:/afile' in a directory for which getwd() would return a path with a trailing separator (largely cosmetic, as reasonable file systems handle such a path correctly). (Perhaps what was meant by PR#14078.)

- unsplit(drop = TRUE) applied to a data frame failed to pass drop to the computation of row names. (PR#14084)

- The "difftime" method of mean() ignored its na.rm argument.

- tcltk::tk_select.list() is now more likely to remove the widget immediately after selection is complete.

- Adding/subtracting a "difftime" object to/from a "POSIXt" or "Date" object works again (it was broken by the addition of Ops.difftime).

- Conversion to latex of an 'Rd' file with no aliases failed.

- wilcox.test(*, conf.int = TRUE) has achieved.level corrected and, for exact = FALSE, now gives an "estimate" which does not depend on the alternative used.

- help.search() failed when the package argument was specified. (PR#14113)

R 2.10 changes

Installation

- configure will be able to find a usable libtiff in some rare circumstances where it did not previously (where libtiff needed to be linked explicitly against -ljpeg).

- cairo >= 1.2 is now required (1.2.0 was released in July 2006) for cairo-based graphics devices (which remain optional).

- A suitable iconv() is now required: support for configure option '--without-iconv' has been withdrawn (it was deprecated in R 2.5.0).

- Perl is no longer "essential". R can be built without it, but scripts R CMD build, check, Rprof and Sd2d require it.

- A system *glob* function is now essential (a working Sys.glob() has been assumed since R 2.9.0 at least).

- C99 support for MBCS is now required, and configure option '--disable-mbcs' has been withdrawn.

- Having a version of tar capable of automagically detecting compressed archives is useful for `utils::untar()`, and so gtar (a common name for GNU tar) is preferred to tar: set environment variable `TAR` to specify a particular `tar` command.

Deprecated & defunct

- The `extended` argument is deprecated in `strsplit()`, `grep()`, `grepl()`, `sub()`, `gsub()`, `regexpr()` and `gregexpr()` (not just the value `extended = FALSE`) and will be removed in R 2.11.0.

User-visible changes

- Package help is now converted from Rd by the R-based converters that were first introduced in 2.9.0. This means

  - Packages that were installed by R-devel after 2009-08-09 should not be used with earlier versions of R, and most aspects of package help (including the runnable examples) will be missing if they are so used.

  - Text, HTML and latex help and examples for packages installed under the new system are converted on-demand from stored parsed 'Rd' files. (Conversions stored in packages installed under R < 2.10.0 are used if no parsed 'Rd' files are found. It is recommended that such packages be re-installed.)

- HTML help is now generated dynamically using an HTTP server running in the R process and listening on the loopback interface.

  - Those worried about security implications of such a server can disable it by setting the environment variable `R_DISABLE_HTTPD` to a non-empty value. This disables `help.start()` and HTML help (so text help is shown instead).

  - The Java/Javascript search engine has been replaced by an HTML interface to `help.search()`. `help.start()` no longer has an argument `searchEngine` as it is no longer needed.

  - The HTML help can now locate cross-references of the form `\link[pkg]{foo}` and `\link[pkg:foo]{bar}` where `foo` is an alias in the package, rather than the documented (basename of a) filename (since the documentation has been much ignored).

New features

- `polygon()`, `pdf()` and `postscript()` now have a parameter `fillOddEven` (default `FALSE`), which controls the mode used for polygon fills of self-intersecting shapes.

- New `debugonce()` function; further, `getOption("deparse.max.lines")` is now observed when debugging, from a code suggestion by John Brzustowski. (PR#13647/8)

- `plot()` methods for `"stepfun"` and hence `"ecdf"` no longer plot points by default for n >= 1000.

- `{[g]sub(*, perl=TRUE)}` now also supports '\E' in order to *end* `\U` and `\L` case changes, thanks to a patch from Bill Dunlap.

- `factor()`, `'levels<-'()`, etc, now ensure that the resulting factor levels are unique (as was always the implied intention). Factors with duplicated levels are still constructible by low-level means, but are now declared illegal.

- New `print()` (S3) method for class "function", also used for auto-printing. Further, .Primitive functions now print and auto-print identically. The new method is based on code suggestions by Romain François.

- The `print()` and `toLatex()` methods for class "sessionInfo" now show the locale in a nicer format and have arguments to suppress locale information.

- In addition to previously only `round()`, there are other 'Math' group (S3) methods for `"difftime"`, such as `floor()`, `signif()`, `abs()`, etc.

- For completeness, `old.packages()` and `available.packages()` allow `type` to be specified (you could always specify `available` or `contriburl`).

- `available.packages()` by default only returns information on the latest versions of packages whose version requirements are satisified by the currently running R.

- `tools::write_PACKAGES()` has a new argument `latestOnly`, which defaults to `TRUE` when only the latest versions in the repository will be listed in the index.

- `getOption()` has a new argument `default` that is returned if the specified option is not set. This simplifies querying a value and checking whether it is `NULL` or not.

- `parse()` now warns if the requested encoding is not supported.

- The `"table"` method of `as.data.frame()` gains a `stringsAsFactors` argument to allow the classifying factors to be returned as character vectors rather than the default factor type.

- If `model.frame.default()` encounters a character variable where `xlev` indicates a factor, it now converts the variable to a factor (with a warning).

- `curve()` now returns a list containing the points that were drawn.

- `spineplot()` now accepts `axes = FALSE`, for consistency with other functions called by `plot.factor()`.

- The Kendall and Spearman methods of `cor.test()` can optionally use continuity correction when not computing exact p-values. (The Kendall case is the wish of PR#13691.)

- R now keeps track of line numbers during execution for code sourced with `options(keep.source = TRUE)`. The source reference is displayed by debugging functions such as `traceback()`, `browser()`, `recover()`, and `dump.frames()`, and is stored as an attribute on each element returned by `sys.calls()`. [Experimental]

- More functions now have an implicit (S4) generic definition.

- `quantile.default()` now disallows factors (wish of PR#13631) and its help documents what numeric-like properties its input need to have to work correctly.

- `weighted.mean()` is now generic and has `"Date"`, `"POSIXct"` and `"POSIXlt"` methods.

- Naming subscripts (e.g. `x[i=1, j=2]`) in `"data.frame"` methods for `[` and `[[` now gives a warning. (Names are ignored in the default method, but could have odd semantics for other methods, and do for the `"data.frame"` ones.)

- `as.data.frame()` has an `"aovproj"` method. (Wish of PR#13505)

- `as.character(x)` for numeric x no longer produces strings such as `"0.30"`, i.e., with trailing zeros. This change also renders levels construction in `factor()` more consistent.

- `codocClasses()`, which checks consistency of the documentation of S4 class slots, now does so in considerably more cases. The documentation of inherited slots (from superclasses) is now optional. This affects '`R CMD check <pkg>`' when the package defines S4 classes.

- `codoc()` now also checks S4 methods for code/documentation mismatches.

- `for()`, `while()`, and `repeat()` loops now always return `NULL` as their (invisible) value. This change was needed to address a reference counting bug without creating performance penalties for some common use cases.

- The `print()` method for `ls.str()` results now obeys an optional `digits` argument.

- The `method` argument of `glm()` now allows user-contributed methods.

- More general `reorder.default()` replaces functionality of `reorder.factor()` and `reorder.character()`.

- The function `aspell()` has been added to provide an interface to the Aspell spell-checker.

- Filters `RdTextFilter()` and `SweaveTeXFilter()` have been added to the tools package to provide support for `aspell()` or other spell checkers.

- `xtabs()` with the new option `sparse = TRUE` now returns a sparse Matrix, using package **Matrix**.

- `contr.sum()` etc gain an argument `sparse` which allows sparse matrices to be returned.

  `contrasts()` also gains a `sparse` argument which it passes to the actual contrast function if that has a formal argument `sparse`.

  `contrasts(f, .) <- val` now also works when `val` is a sparse Matrix. It is planned that `model.matrix()` will work with such factors '`f`' in the future.

- `readNEWS()` will recognize a UTF-8 byte-order mark (BOM) in the 'NEWS' file. However, it is safer to use only ASCII code there because not all editors recognize BOMs.

- New utility function `inheritedSlotNames()` for S4 class programming.

- `tabulate()` now allows `NA`s to pass through (and be ignored).

- If `debug()` is called on an S3 generic function then all methods are debugged as well.

- Outlier symbols drawn by `boxplot()` now obey the `outlwd` argument. Reported by Jurgen Kluge.

- `svd(x)` and `eigen(x)` now behave analogously to `qr(x)` in accepting logical matrices x.

- File 'NEWS' is now in UTF-8, and has a BOM (often invisible) on the first line, and Emacs local variables set for UTF-8 at the end. `RShowDoc("NEWS")` should display this correctly, given suitable fonts.

- `terms.formula(simplify = TRUE)` now does not deparse the LHS and so preserves non-standard responses such as `` `a: b` `` (requested by Sundar Dorai-Raj).
- New function `news()` for building and querying R or package news information.
- `z^n` for integer n and complex z is more accurate now if $|n| <= 65536$.
- `factor(NULL)` now returns the same as `factor(character(0))` instead of an error, and `table(NULL)` consequently does analogously.
- `as.data.frame.vector()` (and its copies) is slightly faster by avoiding a copy if there are no names (following a suggestion of Tim Hesterberg).
- `writeLines()`, `writeBin()` and `writeChar()` have a new argument `useBytes`. If false, character strings with marked encodings are translated to the current locale (as before) but if true they are written byte-by-byte.
- `iconv()` has a new argument `mark` which can be used (by experts) to suppress the declaration of encodings.
- 'DESCRIPTION' 'LinkingTo' specs are now recognized as installation dependencies, and included in package management computations.
- Standardized 'DESCRIPTION' 'License' specs are now available for package management computations.
- `"\uxxxx"` and `"\Uxxxxxxxx"` escapes can now be parsed to a UTF-8 encoded string even in non-UTF-8 locales (this has been implemented on Windows since R 2.7.0). The semantics have been changed slightly: a string containing such escapes is always stored in UTF-8 (and hence is suitable for portably including Unicode text in packages).
- New `as.raw()` method for `"tclObj"` objects (wish of PR#13578).
- 'Rd.sty' now makes a better job of setting email addresses, including using a monospaced font.
- `textConnection()` gains an `encoding` argument to determine how input strings with marked encodings will be handled.
- '`R CMD Rd2pdf`' is available as a shortcut for '`R CMD Rd2dvi --pdf`'.
- '`R CMD check`' now checks links where a package is specified (`\link[pkg]{file}` or `\link[pkg:file]{topic}`), if the package is available. It notes if the package is not available, as in many cases this is an error in the link.
- `identical()` gains three logical arguments, which allow for even more differentiation, notably '`-0`' and '`0`'.
- `legend()` now can specify the `border` color of filled boxes, thanks to a patch from Frederic Schutz.
- Indexing with a vector index to `[[]]` has now been extended to all recursive types.
- Pairlists may now be assigned as elements of lists. (Lists could always be created with pairlist elements, but `[[<-` didn't support assigning them.)
- The parser now supports C-preprocessor-like `#line` directives, so error messages and source references may refer to the original file rather than an intermediate one.
- New functions `findLineNum()` and `setBreakpoint()` work with the source references to find the location of source lines and set breakpoints (using `trace()`) at those lines.
- Namespace importing is more careful about warning on masked generics, thanks to a patch by Yohan Chalabi.
- `detach()` now has an argument `character.only` with the same meaning as for `library()` or `require()`.
- `available.packages()` gains a `filters` argument for specifying the filtering operations performed on the packages found in the repositories. A new built-in `license/FOSS` filter only retains packages for which installation can proceed solely based on packages which can be verified as Free or Open Source Software (FOSS) employing the available license specifications.
- In registering an S3 class by a call to `setOldClass()`, the data part (e.g., the object type) required for the class can be included as one of the superclasses in the Classes argument.
- The argument `f` to `showMethods()` can be an expression evaluating to a generic function, allowing methods to be shown for non-exported generics and other nonstandard cases.
- `sprintf()` now supports `%o` for octal conversions.
- New function `Sys.readlink()` for information about symbolic links, including if a file is a symbolic link.

- Package **tools** has new functions `checkRdaFiles()` and `resaveRdaFiles()` to report on the format of '.rda'/'.RData' data files, and to re-save them in a different compressed format, including choosing the most compact format available.

  A new `INSTALL` option, '`--resave-data`', makes use of this.

- File '~/.R/config' is used in preference to '~/.Rconfig', and these are now documented in 'R Installation and Administration'.

- Logic operations with complex numbers now work, as they were always documented to, and as in S.

- `arrows()` and `segments()` allow one of `x1` or `y1` to be omitted to simplify the specification of vertical or horizontal lines (suggestion of Tim Hesterberg).

- `approxfun()` is faster by avoiding repeated `NA` checks (diagnosis and patch by Karline Soetaert & Thomas Petzoldt).

- There are the beginnings of a Nynorsk translation by Karl Ove Hufthammer.

- `stripchart()` allows par `bg` to be passed in for the background colour for `pch = 21` (wish of PR#13984).

- New generic function `.DollarNames()` to enable class authors to customize completion after the $ extractor.

- `load()`, `save()`, `dput()` and `dump()` now open a not-yet-open connection in the appropriate mode (as other functions using connections directly already did).

Regular expressions

- A different regular expression engine is used for basic and extended regexps and is also for approximate matching. This is based on the TRE library of Ville Laurikari, a modifed copy of which is included in the R sources.

  This is often faster, especially in a MBCS locale.

  Some known differences are that it is less tolerant of invalid inputs in MBCS locales, and in its interpretation of undefined (extended) regexps such as `"^*"`. Also, the interpretation of ranges such as [W-z] in caseless matching is no longer to map the range to lower case.

  This engine may in future be used in 'literal' mode for `fixed = TRUE`, and there is a compile-time option in 'src/main/grep.c' to do so.

- The use of repeated boundary regexps in `gsub()` and `gregexpr()` as warned about in the help page does not work in this engine (it did in the previous one since 2005).

- Extended (and basic) regexps now support same set of options as for `fixed = TRUE` and `perl = TRUE`, including `useBytes` and support for UTF-8-encoded strings in non-UTF-8 locales.

- `agrep()` now has full support for MBCS locales with a modest speed penalty. This enables `help.search()` to use approximate matching character-wise rather than byte-wise.

- `[g]sub` use a single-pass algorithm instead of matching twice and so is usually faster.

- The `perl = TRUE` versions now work correctly in a non-UTF-8 MBCS locale, by translating the inputs to UTF-8.

- `useBytes = TRUE` now inhibits the translation of inputs with marked encodings.

- `strsplit()` gains a `useBytes` argument.

- The algorithm used by `strsplit()` has been reordered to batch by elements of `split`: this can be much faster for `fixed = FALSE` (as multiple compilation of regexps is avoided).

- The help pages, including `?regexp`, have been updated and should be consulted for details of the new implementations.

Help & 'Rd' file changes

- A new dynamic HTML help system is used by default, and may be controlled using `tools::startDynamicHelp()`. With this enabled, HTML help pages will be generated on request, resolving links by searching through the current `.libPaths()`. The user may set `options("help.ports")` to control which IP port is used by the server.

- `help.start()` no longer sets `options(htmlhelp = TRUE)` (it used to on Unix but not on Windows). Nor does it on Unix reset the `"browser"` option if given an argument of that name.

  Arguments `update` and `remote` are now available on all platforms: the default is `update = FALSE` since the http server will update the package index at first use.

- `help()` has a new argument `help_type` (with default set by the option of that name) to supersede `offline`, `htmlhelp` and `chmhelp` (although for now they still work if `help_type` is unset). There is a new type, `"PDF"` to allow offline PDF (rather than PostScript).

A function `offline_help_helper()` will be used if this exists in the workspace or further down the search path, otherwise the function of that name in the 'utils' name space is used.

- Plain text help is now used as the fallback for HTML help (as it always was for Compiled HTML help on Windows).
- It is possible to ask for static HTML pages to be prebuilt via the configure option '`--enable-prebuilt-html`'. This may be useful for those who wish to make HTML help available outside R, e.g. on a local web site.
- An experimental tag `\Sexpr` has been added to 'Rd' files, to evaluate expressions at build, install, or render time. Currently install time and render time evaluation are supported.
- Tags `\if`, `\ifelse` and `\out` have been added to allow format-specific (or more general, using `\Sexpr`) conditional text in man pages.
- The `parse_Rd()` parser has been made more tolerant of coding errors in 'Rd' files: now all syntax errors are reported as warnings, and an attempt is made to continue parsing.
- `parse_Rd()` now has an argument `fragment` (default `FALSE`) to accept small fragments of 'Rd' files (so that `\Sexpr` can output Rd code which is then parsed).
- `parse_Rd()` now always converts its input to UTF-8. The Rd2* rendering functions have a new parameter, `outputEncoding`, which controls how their output is encoded.
- `parse_Rd()` no longer includes the newline as part of a "%"-style comment.
- There have been various bug fixes and code reorganization in the Rd renderers Rd2HTML, Rd2latex, Rd2txt, and Rd2ex.

  All example files are now created with either ASCII or UTF-8 encoding, and the encoding is only marked in the file if there is any non-UTF-8 code (previously it was marked if the help file had non-ASCII contents, possibly in other sections).
- `print.Rd()` now adds necessary escape characters so that printing and re-parsing an Rd object should produce an equivalent object.
- `parse_Rd()` was incorrectly handling multiple backslashes in R code strings, converting 4n+3 backslashes to 2n+1 instead of 2n+2.
- `parse_Rd()` now recognizes the *t*ag within a quoted string in R-like text.
- `parse_Rd()` now treats the argument of `as` LaTeX-like, rather than verbatim.

Compression

- New function `untar()` to list or unpack tar archives, possibly compressed. This uses either an external `tar` command or an internal implementation.
- New function `tar()` to create (possibly compressed) tar archives.
- New functions `memCompress()` and `memDecompress()` for in-memory compression and decompression.
- `bzfile()` has a `compress` argument to select the amount of effort put into compression when writing.
- New function `xzfile()` for use with xz-compressed files. (This can also read files compressed by some versions of 'lzma'.)
- `gzfile()` looks at the file header and so can now also read bzip2-ed files and xz-compressed files.
- There are the new options of `save(compress = "bzip2")` and "xz" to use bzip2 or xz compression (which will be slower, but can give substantially smaller files). Argument compression_level gives finer control over the space/time tradeoffs.

  `load()` can read such saves (but only as from this version of R).
- `R CMD INSTALL/check` and `writePACKAGES` (**tools**) accept a wider range of compressed tar archives. Precisely how wide depends on the capabilities of the host system's `tar` command: they almost always include '.tar.bz2' archives, and with modern versions of tar other forms of compression such as lzma and xz, and arbitrary extensions.
- '`R CMD INSTALL`' has a new option '`--data-compress`' to control the compression used when lazy-loading data. New possibilities are '`--data-compress=bzip2`' which will give ca 15% better compression at the expense of slower installation times, and '`--data-compress=xz`', often giving even better compression on large datasets at the expense of much longer installation times. (The latter is used for the recommended packages: it is particularly effective for **survival**.)
- `file()` for `open = ""`, `"r"` or `"rt"` will automagically detect compressed files (from gzip, bzip2 or xz). This means that compressed files can be specified by file name (rather than via a `gzfile()` connection) to `read.table()`, `readlines()`, `scan()` and so on.

- `data()` can handle compressed text files with extensions .{txt,tab,csv}.{gz,bz2,xz} .

Deprecated & defunct

- `png(type="cairo1")` is defunct: the value is no longer recognized.
- `tools::Rd_parse()` is defunct (as this version of R uses only Rd version 2).
- Use of '~/.Rconf' (which was deprecated in favour of '~/.Rconfig' in 2004) has finally been removed.
- Bundles of packages are deprecated. See 'Writing R Extensions' for the steps needed to unbundle a bundle.
- `help()` arguments `offline`, `htmlhelp` and `chmhelp` are deprecated in favour of `help_type`.
- `clearNames()` (**stats**) is deprecated for `unname()`.
- Basic regular expressions (`extended = FALSE`) are deprecated in strsplit, grep and friends. There is a precise POSIX standard for them, but it is not what recent RE engines implement, and it seems that in almost all cases package authors intended `fixed = TRUE` when using `extended = FALSE`.
- `methods::trySilent()` is deprecated for `try(*, silent=TRUE)` or - more efficiently and flexibly - something like `tryCatch(*, error = function(e)` e).
- `index.search()` is deprecated: there are no longer directories of types other than 'help'.

Internationalization

- There is some makefile support for adding/updating translations in packages: see 'po/README' and 'Writing R Extensions'.

  There is support for the use of 'dngettext' for C-level translations in packages: see 'Writing R Extensions'.

Bug fixes

- `trigamma(x)` and other `psigamma(x, n)` calls are now accurate for very large `abs(x)`. (PR#14020)
- `[g]sub()` could use excessive stack space when used with a very long vector of non-ASCII data.
- The default method of `weighted.mean(na.rm = TRUE)` did not omit weights for `NA` observations in 2.10.0. (PR#14032)
- Assigning an extra 0-length column to a data frame by `DF[, "foo"] <- value` now works in most cases (by filling with `NA`s) or fails. (It used to give a corrupt data frame.)
- `validObject()` avoids an error during evaluation in the case of various incorrect slot definitions.
- `n:m` now returns a result of type "integer" in a few more boundary cases.
- The `zap.ind` argument to `printCoefmat()` did not usually work as other code attempted to ensure that non-zero values had a non-zero representation.
- `printCoefmat()` formatted groups of columns together, not just the `cs.ind` group but also the `zap.ind` group and a residual group. It now formats all columns except the cs.ind group separately (and zaps the `zap.ind` group column-by-column). The main effect will be see in the output from print.anova, as this grouped SS-like columns in the `zap.ind` group.
- `R_ReplDLLinit()` initializes the top-level jump so that some embedded applications on Windows no longer crash on error.
- `identical()` failed to take the encoding of character strings into account, so identical byte patterns are not necessarily identical strings, and similarly Latin-1 and UTF-8 versions of the same string differ in byte pattern.
- `methods(f)` used to warn unnecessarily for an S4 generic `f` which had been created based on an existing S3 generic.
- The check for consistent ordering of superclasses was not ignoring all conditional relations (the symptom was usually spurious warnings for classes extending "array").
- Trying to assign into a raw vector with an index vector containing `NA`s could cause a segfault. Reported by Hervé Pagès.
- Rscript could segfault if (by user error) its filename argument was missing. Reported by Martin Morgan.
- `getAnywhere()` (and functions that use it, including argument completion in the console) did not handle special built-in functions. Reported by Romain François.
- `order()` was missing a `PROTECT()` call and so could segfault when called on character data under certain (rare) circumstances involving marked non-native encodings.

- `prettyNum(z, drop0trailing=TRUE)` did not work correctly when z was a complex vector. Consequently, `str(z, ...)` also did not. (PR#13985)

- `make distclean` removed too many files in etc/ if `builddir = srcdir`.

- '`R CMD`' replaced `TEXINPUTS` rather than appending to it (as documented and intended).

- `help.start()` no longer fails on unix when "browser" is a function.

- `pbeta(x, *, log.p = TRUE)` is sometimes more accurate, e.g., for very small x.

- Unserializing a pre-2.8 workspace containing pure ASCII character objects with a LATIN1 or UTF-8 encoding would corrupt the CHARSXP cache.

Changes in R 2.9.2 patched

New features

- On systems using ICU for collation (including Mac OS X), using `Sys.setlocale()` to change the LC_COLLATE setting is more likely to change the collation provided by ICU.

Bug fixes

- Names of datasets could clash with temporary filenames used when running examples, causing errors.

- `as.complex()` sometimes warned about `NA`s on coercions and sometimes not (when the C function asComplex was used, e.g. on list elements). (PR#13942)

- `cat()` on an unopened connection could close it twice, and `file()` connections segfaulted on some systems.

- Printing a list could segfault if the elements are nested too deeply.

Changes in R 2.9.2

New features

- `install.packages(NULL)` now lists packages only once even if they occur in more than one repository (as the latest compatible version of those available will always be downloaded).

- `approxfun()` and `approx()` now accept a `rule` of length two, for easy specification of different interpolation rules on left and right.

  They no longer segfault for invalid zero-length specification of `yleft`, `yright`, or `f`.

- `seq_along(x)` is now equivalent to `seq_len(length(x))` even where `length()` has an S3/S4 method; previously it (intentionally) always used the default method for `length()`.

- PCRE has been updated to version 7.9 (for bug fixes).

- `agrep()` uses 64-bit ints where available on 32-bit platforms and so may do a better job with complex matches. (E.g. PR#13789, which failed only on 32-bit systems.)

Deprecated & defunct

- '`R CMD Rd2txt`' is deprecated, and will be removed in 2.10.0. (It is just a wrapper for '`R CMD Rdconv -t txt`'.)

- `tools::Rd_parse()` is deprecated and will be removed in 2.10.0 (which will use only Rd version 2).

Bug fixes

- `parse_Rd()` still did not handle source reference encodings properly.

- The C utility function PrintValue no longer attempts to print attributes for CHARSXPs as those attributes are used internally for the CHARSXP cache. This fixes a segfault when calling it on a CHARSXP from C code.

- PDF graphics output was producing two instances of anything drawn with the symbol font face. (Report from Baptiste Auguié.)

- `length(x) <-` newval and `grep()` could cause memory corruption. (PR#13837)

- If `model.matrix()` was given too large a model, it could crash R. (PR#13838, fix found by Olaf Mersmann.)

- `gzcon()` (used by `load()`) would re-open an open connection, leaking a file descriptor each time. (PR#13841)

- The checks for inconsistent inheritance reported by `setClass()` now detect inconsistent superclasses and give better warning messages.

- `print.anova()` failed to recognize the column labelled `P(>|Chi|)` from a Poisson/binomial GLM anova as a p-value column in order to format it appropriately (and as a consequence it gave no significance stars).
- A missing PROTECT caused rare segfaults during calls to `load()`. (PR#13880, fix found by Bill Dunlap.)
- `gsub()` in a non-UTF-8 locale with a marked UTF-8 input could in rare circumstances overrun a buffer and so segfault.
- 'R CMD Rdconv --version' was not working correctly.
- Missing PROTECTs in `nlm()` caused "random" errors. (PR#13381 by Adam D.I. Kramer, analysis and suggested fix by Bill Dunlap.)
- Some extreme cases of `pbeta(log.p = TRUE)` are more accurate (finite values < -700 rather than -Inf). (PR#13786)

  `pbeta()` now reports on more cases where the asymptotic expansions lose accuracy (the underlying TOMS708 C code was ignoring some of these, including the PR#13786 example).
- `new.env(hash = TRUE, size = NA)` now works the way it has been documented to for a long time.
- `tcltk::tk_choose.files(multi = TRUE)` produces better-formatted output with filenames containing spaces. (PR#13875)
- 'R CMD check --use-valgrind' did not run valgrind on the package tests.
- The `tclvalue()` and the `print()` and `as.xxx` methods for class `"tclObj"` crashed R with an invalid object – seen with an object saved from an earlier session.
- 'R CMD BATCH' garbled options '-d' <debugger> (useful for valgrind, although '--debugger=valgrind' always worked)
- `INSTALL` with LazyData and Encoding declared in 'DESCRIPTION' might have left `options("encoding")` set for the rest of the package installation.

Changes in R 2.9.1

New features

- New function `anyDuplicated(x)` returns 0 (= `FALSE`) or the index of the first duplicated entry of x.
- `matplot()`, `matlines()` and `matpoints()` now also obey a `lend` argument, determining line end styles. (Wish of PR#13619).
- `bw.SJ()`, `bw.bcv()` and `bw.ucv()` now gain an optional `tol` argument allowing more accurate estimates.
- `new.packages()` no longer regards packages with the same name as a member of an installed bundle as 'new' (this is now consistent with the dependency checks in `install.packages()`).

  It no longer reports on partially installed bundles (since members can be updated individually if a bundle is unbundled).
- `old.packages()` and hence `updates.packages()` will look for updates to members of package bundles before updates to the whole bundle: this allow bundles to be split and installations updated.
- `nlminb()` gives better non-convergence messages in some cases.
- S3 method dispatch will support S4 class inheritance for S3 methods, for primitives and via `UseMethod()`, if the argument `S3methods=TRUE` is given to `setClass()`. S4 method dispatch will use S3 per-object inheritance if `S3Class()` is set on the object. See `?Methods` and the paper referenced there.
- 'R CMD INSTALL' is more tolerant of (malformed) packages with a 'man' directory but no validly named '.Rd' files.
- 'R CMD check' now reports where options are used that cause some of the checks to be skipped.
- `RSiteSearch` has been updated to be consistent with the new layout of the search site itself, which now includes separate options for vignettes, views, and r-sig-mixed-models, as well as changed names for r-help. (Contributed by Jonathan Baron.)
- That 'R CMD check' makes use of a '<pkg>/tests/Examples/<pkg>-Ex.Rout.save' file as a reference result is now documented in 'Writing R Extensions'.

Deprecated & defunct

- `print.atomic()` (defunct since 1.9.0) has been removed since it caused confusion for an S4 class union "atomic".
- `png(type="cairo1")` is deprecated – it was only needed for platforms with 1.0 <= cairo < 1.2.

Bug fixes

- The '...' argument was not handled properly when '...' was found in the enclosure of the current function, rather than in the function header itself. (This caused `integrate()` to fail in certain cases.)
- `interaction()` now ensures that the levels of the result are unique.
- `packageDescription()` and hence `sessionInfo()` now report the correct package version also for a non-attached loaded namespace of a version different from the default `lib.loc`.
- `smoothScatter()` now also works when e.g. `xlim[2] < xlim[1]`.
- `parse_Rd()` would mishandle braces when they occurred at the start of a line within an R string in an 'Rd' file (reported by Alex Couture-Beil) or when they occurred in an R comment (reported by Mark Bravington).
- `readNEWS()` missed version numbers with more than one digit.
- building R '`--without-x`' no longer fails (PR#13665)
- `printCoefmat(cbind(0,1))` now works too (PR#13677)
- `bw.SJ(c(1:99, 1e6))` now works too.
- `Rd2txt()` could not handle empty descriptions of items in an 'Rd' file (reported by Mark Bravington), and did not wrap long lists of arguments if they were given in a single item.
- `stars()` would do a partial plot when called with `plot = FALSE`; it now consistently returns the locations of the stars.
- `Rd2latex()` could not handle empty sections.
- `old.packages()` and hence `update.packages()` would fail on a repository which contained only one package but with multiple versions of that package.
- `as.character.Rd()` added extra braces when displaying two-argument macros. (Report and fix by Manuel Eugster.)
- `unsplit()` was misbehaving in the case of single-column data frames. (Reported by Will Gray.)
- `as(I(1), "vector")` and similar coercions from objects of "unregistered" S3 classes now work.
- srcref records produced by `parse()` and `parse_Rd()` did not record the encoding of the source file. (Reported by Romain François.)
- The X11 version of `View()` was misbehaving in MBCS locales, and PgUp/PgDn now behave better, thanks to a patch from Ei-ji Nakama.
- '`R CMD check`' looked at the environment variable `PDFLATEX`, but as from R 2.8.1 '`R CMD Rd2dvi`' used `R_PDFLATEXCMD` in preference, and that was set by '`R CMD`' (and not `PDFLATEX`). Now '`R CMD check`' looks at `R_PDFLATEXCMD`.
- The new (in 2.9.0) `stringsAsFactors` argument to `expand.grid()` was not working: it now does work but has default `TRUE` for backwards compatibility.
- `tcrossprod(<1d-array>, <matrix>)` now does work when the arguments are of compatible dimensions.
- `qbinom()` now is accurate also in (large size, small prob) cases. (PR#13711)
- The calculation of the Spearman p-value in `cor.test()` is slightly more accurate in some cases. (PR#13574)
- The `digamma()`, `trigamma()` and `psigamma()` functions could be inaccurate for values of x around 1e-15 due to cancellation. (PR#13714).
- `median.default()` was altered in 2.8.1 to use `sum()` rather than `mean()`, although it was still documented to use `mean()`. This caused problems for POSIXt objects, for which `mean()` but not `sum()` makes sense, so the change has been reverted.
- Assigning an extra 0-length column to a data frame by `DF$foo <- value` gave a corrupt data frame rather than failing. (PR#13724) This also happened for `DF[["foo"]] <- value`.
- '`R CMD INSTALL`' no longer gives a spurious warning about old R versions ignoring multiple dependencies, if the conditions are known to be satisfied.
- The test for setting `dim()` allowed a value with two or more `NA`s to be set on a 0-length object. (PR#13729) Also, it allowed an even number of negative values.
- `xtfrm()`, `rank()`, `sort()` and `order()` did not always make use of custom comparison methods specific to the class of elements being sorted.
- Increase NAMED value on `seq` value in `for()` loop so loop code cannot modify `seq` value.
- Prevent rectangles of size < 0.5 pixel from disappearing in Quartz when using rastered backend. (PR#13744)

- Printing '_NA_complex_' had a low-level thinko; patch thanks to Bill Dunlap.
- CP1257 encoding for postscript/PDF has been corrected. (PR#13736)
- aov() with an error term was evaluating the ... arguments in 2.9.0 whilst checking their names, so could fail by evaluating them in the wrong place. (PR#13733)
- The print() method for arima() failed if all coefs were fixed.
- 'R CMD INSTALL --no-latex' was not implemented in 2.9.0 (only).
- Added a needed PROTECT call in RunFinalizers to handle cases where the routine is called recursively from a GC in a finalizer.
- Constructing error messages about unused arguments in calls to closures no longer evaluates the arguments.
- qr(x, LAPACK=TRUE) did not coerce integer x to numeric.
- qr.coef() misbehaved in the LAPACK case with a matrix RHS, so that solve(qr(x, LAPACK=TRUE)) gave wrong results. (Found by Avraham Adler, PR#13762 by Ravi Varadhan.)

Changes on CRAN

by Kurt Hornik and Achim Zeileis

New CRAN task views

ClinicalTrials Design, Monitoring, and Analysis of Clinical Trials.
Packages: **ClinicalRobustPriors***, **GroupSeq***, **HH**, **Hmisc***, **MChtest***, **PwrGSD***, **asypow**, **bifactorial***, **binomSamSize**, **blockrand***, **clinfun***, **coin**, **copas**, **epibasix**, **epicalc**, **experiment***, **gsDesign***, **ldbounds***, **meta**, **metafor**, **multcomp**, **rmeta**, **seqmon***, **speff2trial***, **ssanv**, **survival***.
Maintainer: Ed Zhang.

MedicalImaging Medical Image Analysis.
Packages: **AnalyzeFMRI***, **DICOM***, **PET***, **Rniftilib***, **VR**, **adimpro***, **bitops**, **dcemri***, **dti***, **fmri***, **minpack.lm**, **tractor.base***.
Maintainer: Brandon Whitcher.

(* = core package)

New packages in CRAN task views

Bayesian **BAS**, **glmmBUGS**.

ChemPhys **AquaEnv**, **CHNOSZ**, **NMRS**, **gpls**, **nlreg**, **paltran**, **plspm**, **quantchem**, **spls**.

Cluster **nnclust**.

Distributions **Lmoments**, **VarianceGamma**, **denstrip**, **evdbayes**, **evir**, **fExtremes**, **fitdistrplus**, **gamlss.dist***, **gamlss.mx**, **ig**.

Econometrics **CADFtest**, **Mcomp**, **Zelig**, **aod**, **expsmooth**, **fma**, **forecast**.

Environmetrics **aod**, **primer**, **vegetarian**.

ExperimentalDesign **DoE.base***, **DoE.wrapper***, **RcmdrPlugin.DoE**.

Finance **YieldCurve**, **atmi**, **forecast**, **ttrTests**.

HighPerformanceComputing **cudaBayesreg**, **doMC**, **doSNOW**, **foreach**, **speedglm**.

MachineLearning **penalizedSVM**.

NaturalLanguageProcessing **openNLPmodels.en**, **openNLPmodels.es**.

Optimization **Rcsdp**, **nleqslv**.

Psychometrics **MLCM**, **SEMModComp**, **irtProb**, **latdiag**.

Spatial **RgoogleMaps**, **nlme**.

Survival **AER**, **BMA**, **BayHaz**, **DAAG**, **Epi**, **ICE**, **LearnBayes**, **LogicReg**, **MAMSE**, **MCMCglmm**, **MCMCpack**, **OrdFacReg**, **SMIR**, **SMPracticals**, **SimHap**, **VGAM**, **XReg**, **clinfun**, **cmprskContin**, **coin**, **condGEE**, **coxme**, **epiR**, **etm**, **fitdistrplus**, **gbm**, **gof**, **gss**, **interval**, **ipred**, **km.ci**, **kmi**, **lmec**, **locfit**, **logspline**, **maxstat**, **mixAK**, **mstate**, **multcomp**, **multtest**, **mvpart**, **nltm**, **p3state.msm**, **pamr**, **party**, **penalized**, **phmm**, **polspline**, **quantreg**, **rankhazard**, **rhosp**, **risksetROC**, **rms***, **spatstat**, **superpc**, **survcomp**, **survey**, **uniCox**.

TimeSeries **CADFtest**, **EvalEst**, **KFAS**, **Mcomp**, **dlnm**, **expsmooth**, **fma**, **forecast***, **fractalrock**, **mar1s**, **tiger**, **tis**.

(* = core package)

New contributed packages

ADGofTest Anderson-Darling GoF test with p-value calculation based on Marsaglia's 2004 paper "Evaluating the Anderson-Darling Distribution". By Carlos J. Gil Bellosta.

AGSDest Estimation in adaptive group sequential trials. By Niklas Hack and Werner Brannath.

CDFt Statistical downscaling through CDF transform. Also performs computation of the Cramèr-von Mises and Kolmogorov-Smirnov statistics. By Mathieu Vrac and Paul-Antoine Michelangeli.

COP Variables selection for index models via correlation pursuit. By Wenxuan Zhong.

COREelearn A machine learning suite for classification, regression, feature evaluation and ordinal evaluation, based on C++ code. Contains several model learning techniques in classification and regression, for example decision and regression trees with optional constructive induction and models in the leafs, random forests, kNN, naive Bayes, and locally weighted regression. Especially strong in feature evaluation algorithms where it contains several variants of Relief algorithm and many impurity based attribute evaluation functions, e.g., Gini, information gain, MDL, DKM, etc. Additional strengths are its ordEval algorithm and its visualization used for ordinal features and classes. By Marko Robnik-Sikonja and Petr Savicky.

CalciOMatic Simulate and analyze calcium imaging data obtained with ratiometric dyes. By Sebastien Joucla, Christophe Pouzat.

CircNNTSR Statistical analysis of circular data using non-negative trigonometric sums (NNTS) models. Includes functions for calculation of densities and distributions, the estimation of parameters, plotting and more. By Juan José Fernández-Durán and Maria Mercedes Gregorio-Domínguez.

CircSpatial A collection of functions for color continuous high resolution images of circular spatial data, circular kriging, and simulation of circular random fields. By Bill Morphet.

DRI DR-Integrator: Integrative analysis of DNA copy number and gene expression data described in Salari et al (2009). By Keyan Salari, Robert Tibshirani, and Jonathan R. Pollack.

DTDA Doubly Truncated Data Analysis. Implements different algorithms for analyzing randomly, one-sided and two-sided (i.e., doubly) truncated data. By Carla Moreira, Jacobo de Uña-Álvarez and Rosa Crujeiras.

Daim Diagnostic accuracy of classification models. Several functions for evaluating the accuracy of classification models. Provides performance measures 'cv', '0.632' and '0.632+', estimation of the misclassification rate, sensitivity, specificity and AUC. If an application is computationally intensive, parallel execution can be used to reduce the time taken. By Sergej Potapov, Werner Adler and Berthold Lausen.

Deducer An intuitive graphical data analysis system for use with **JGR**. By Ian Fellows.

DiversitySampler Functions for re-sampling a community matrix to compute Shannon's Diversity index at different sampling levels. By Matthew K. Lau.

DoE.base Creation of full factorial experimental designs and designs based on orthogonal arrays for (industrial) experiments. Additionally provides some utility functions used also by other DoE packages. By Ulrike Grömping.

DoE.wrapper Wrapper package for design of experiments functionality. Creates various kinds of designs for (industrial) experiments by using and possibly enhancing design generation routines from other packages. Currently, response surface designs from package **rsm** and latin hypercube samples from package **lhs** have been implemented. By Ulrike Grömping.

EQL Extended Quasi Likelihood function (EQL). Computation of the EQL for a given family of variance functions, Saddlepoint-approximations and related auxiliary functions (e.g., Hermite polynomials). By Thorn Thaler.

ElectroGraph Enhanced routines for plotting and analyzing valued relational data, considering valued ties. In particular, relative distances are calculated using social conductance methods. By Andrew C. Thomas.

EnQuireR A package dedicated to questionnaires. By Fournier Gwenaëlle, Cadoret Marine, Fournier Olivier, Le Poder François, Bouche Jérôme, and Lê Sébastien.

EvalEst Dynamic Systems Estimation (DSE) extensions. By Paul Gilbert.

FEST Identification of family relations using linked markers. By Øivind Skare.

FME A Flexible Modeling Environment for inverse modeling, sensitivity, identifiability, Monte Carlo analysis. Intended to work with models written as a set of differential equations that are solved either by an integration routine from package **deSolve**, or a steady-state solver from package **rootSolve**, but can also be used with other types of functions. By Karline Soetaert and Thomas Petzoldt.

FRB Robust inference based on applying Fast and Robust Bootstrap on robust estimators. Available methods are multivariate regression, PCA and Hotelling tests. By Ella Roelant, Stefan Van Aelst and Gert Willems.

GGMselect Gaussian Graphs Models selection: graph estimation in Gaussian Graphical Models. The main functions return the adjacency matrix of an undirected graph estimated from a data matrix. By Annie Bouvier, Christophe Giraud, Sylvie Huet, and Verzelen N.

GLMMarp Generalized Linear Multilevel Model with AR(p) errors. Functions to estimate the GLMM-AR(p) model for analyzing discrete time-series cross-sectional data via MCMC simulation. Also contains several useful utility functions, including an independent function for computing the Bayes factor with GLMM-AR(p) output, a function to recover the random coefficients at the individual level, and a function to do prediction by using the posterior distributions. By Xun Pang.

GOFSN Goodness-Of-Fit tests for the family of Skew-Normal models. Implements a method for checking if a skew-normal model fits the observed dataset, when all parameters are unknown. While location and scale parameters are estimated by moment estimators, the shape parameter is integrated with respect to the prior predictive distribution, as proposed in Box (1980). A default and proper prior on skewness parameter is used to obtain the

prior predictive distribution, as proposed in Cabras, & Castellanos (2008). By Veronica Paton Romero.

GWAF Genome-Wide Association analyses with Family data: Functions to test genetic associations between SNPs and a continuous/dichotomous trait using family data, and to make genome-wide p-value plots and QQ plots. By Ming-Huei Chen and Qiong Yang.

GeneReg Infer time delay gene regulatory networks using time course gene expression profiles. The main idea of the time delay linear model is to fit a linear regression model using a set of putative regulators to estimate the transcription pattern of a specific target gene. By Tao Huang.

Geneclust Simulation and analysis of spatial structure of population genetics data. By Sophie Ancelet.

HMR Statistical analysis of static chamber concentration data for trace gas flux estimation. By Asger R. Pedersen.

HSAUR2 A Handbook of Statistical Analyses Using R (2nd Edition). Functions, data sets, analyses and examples from the second edition of the book "A Handbook of Statistical Analyses Using R" (Brian S. Everitt and Torsten Hothorn, Chapman & Hall/CRC, 2008). By Brian S. Everitt and Torsten Hothorn.

HTMLUtils Facilitates automated HTML report creation, in particular framed HTML pages and dynamically sortable tables. By Markus Loecher.

HWEBayes Bayesian investigation of Hardy-Weinberg Equilibrium via estimation and testing. Three models are currently considered: HWE, a model parametrized in terms of the allele frequencies and a single inbreeding coefficient f, and the saturated model. Testing is based on Bayes factors. By Jon Wakefield.

Haplin Performs a genetic association analysis of case-parent triad (trio) data with multiple markers. Can also incorporate complete or incomplete control triads, for instance independent control children. Estimation is based on haplotypes, for instance SNP haplotypes, even though phase is not known from the genetic data. Estimates relative risk and p-values associated with each haplotype. Uses MLE to make optimal use of data from triads with missing genotypic data, for instance if some SNPs has not been typed for some individuals. Also allows estimation of effects of maternal haplotypes, particularly appropriate in perinatal epidemiology. By H. K. Gjessing.

HybridMC An implementation of the Hybrid Monte Carlo and Multipoint Hybrid Monte Carlo sampling techniques described in Liu (2001), "Monte Carlo Strategies in Computing". By Richard D. Morey.

IsoGene Testing for monotonic relationship between gene expression and doses in a microarray experiment. Several testing procedures including the global likelihood ratio test (Bartholomew, 1961), Williams (1971, 1972), Marcus (1976), M (Hu et al. 2005) and the modified M (Lin et al. 2007) are used to test for the monotonic trend in gene expression with respect to doses. BH (Benjamini and Hochberg 1995) and BY (Benjamini and Yekutilie 2004) FDR controlling procedures are applied to adjust the raw p-values obtained from the permutations. By Dan Lin et al.

KFAS Functions for fast Kalman filtering, state and disturbance smoothing, forecasting and simulation of multivariate time-variant state space models. All functions can use exact diffuse initialization when distributions of some or all elements of initial state vector are unknown. Filtering, state smoothing and simulation functions use the sequential processing algorithm, which is faster than standard approach, and also allows singularity of the prediction error variance matrix. By Jouni Helske.

LLdecomp Decomposes a set of variables into cliques and separators depending on their association which is measured using Random Forests. By Corinne Dahinden.

MCMChybridGP Hybrid Markov chain Monte Carlo using Gaussian Processes. Uses hybrid MCMC to simulate from a multimodal target distribution. A Gaussian process approximation makes this possible when derivatives are unknown. Serves to minimize the number of function evaluations in Bayesian calibration of computer models using parallel tempering. Allows replacement of the true target distribution in high temperature chains, or complete replacement of the target. Mark J. Fielding.

MLCM Maximum Likelihood Conjoint Measurement. Conjoint measurement is a psychophysical procedure in which stimulus pairs are presented that vary along 2 or more dimensions and the observer is required to compare the stimuli along one of them. This package contains functions to estimate the contribution of the n scales to the judgment by a maximum likelihood method under several hypotheses of how the perceptual dimensions interact. By Kenneth Knoblauch and Laurence T. Maloney.

MMIX Implement different types of model mixing and model selection methods for linear or logistic models. By Marie Morfin and David Makowski.

NeatMap Non-clustered heatmap alternatives. Creates heatmap like plots in 2 and 3 dimensions, without the need for cluster analysis. Like the heatmap, the plots created display both a dimensionally reduced representation of the data as well as the data itself. They are intended to be used in conjunction with dimensional reduction techniques such as PCA. By Satwik Rajaram and Yoshi Oono.

ORIClust Order-restricted Information Criterion-based Clustering of genes. Clusters are given by genes matched to prespecified profiles across various ordered treatment groups. Particularly useful for analyzing data obtained from short time-course or dose-response microarray experiments. By Tianqing Liu, Nan Lin, Ningzhong Shi and Baoxue Zhang.

OjaNP Functions for the Oja median, Oja signs and ranks and methods based upon them. By Daniel Fischer, Jyrki Möttönen, Klaus Nordhausen and Daniel Vogel.

PCIT Partial Correlation Coefficient with Information Theory. Provides the PCIT algorithm developed by Reverter and Chan (2008) which identifies significant gene to gene associations to define edges in a weighted network. By Nathan S. Watson-Haigh.

PCS Calculate the probability of correct selection (PCS). Given k populations (can be in thousands), what is the probability that a given subset of size t contains the true top t populations? This package finds this probability and offers three tuning parameters (G, d, L) to relax the definition. By Jason Wilson.

PLIS Multiplicity control using Pooled LIS (PLIS) statistics. PLIS is a multiple testing procedure for testing several groups of hypotheses. Linear dependency is expected from the hypotheses within the same group and is modeled by hidden Markov Models. It is noted that, for PLIS, a smaller p value does not necessarily imply more significance because of dependency among the hypotheses. A typical application of PLIS is to analyze genome wide association studies datasets, where SNPs from the same chromosome are treated as a group and exhibit strong linear genomic dependency. By Zhi Wei & Wenguang Sun.

PearsonDS Implementation of the Pearson distribution system, including full support for the (d,p,q,r)-family of functions for probability distributions and fitting via method of moments and maximum likelihood method. By Martin Becker.

QCA3 Functions for Qualitative Comparative Analysis (QCA). Can be used for all three types of QCA. Has methods for simplifying assumption and contradictory simplifying assumption, and can return constrained results by including or excluding specific conditions. By Ronggui Huang.

R.filesets Easy handling of and access to files organized in structured directories. A file set refers to a set of files located in one or more directories on the file system. This package provides classes and methods to locate, setup, subset, navigate and iterative over such sets. The API is designed such that these classes can be subsetted to provide for instance a richer API for special file formats. By Henrik Bengtsson.

R2PPT Simple R Interface to Microsoft PowerPoint using **rcom**. By Wayne Jones.

R2wd Write MS-Word documents from R, using the statconnDCOM server to communicate with MS-Word via the COM interface. By Christian Ritter.

RBerkeley Interface to Embedded Oracle Berkeley DB(tm). By Jeffrey A. Ryan.

RC Reproducible Computing. Allows the user to create and use reproducible computations for the purpose of research and education. The meta data about the computations are stored in a remote repository which is hosted at `www.freestatistics.org`. By Patrick Wessa.

REEMtree Estimates regression trees with random effects as a way to use data mining techniques to describe longitudinal or panel data. By Rebecca Sela and Jeffrey Simonoff.

RImageJ Bindings between R and the ImageJ Java based image processing and analysis platform. By Romain François and Philippe Grosjean.

RInside C++ classes to embed R in C++ applications. By Dirk Eddelbuettel.

RLastFM R interface to last.fm API. By Greg Hirson.

RPMM Recursively Partitioned Mixture Model for Beta and Gaussian Mixtures. This is a model-based clustering algorithm that returns a hierarchy of classes, similar to hierarchical clustering, but also similar to finite mixture models. By E. Andres Houseman.

RSiena Simulation Investigation for Empirical Network Analysis. Fits models to longitudinal networks. By various authors.

Rassoc Robust tests for case-control genetic association studies: allelic based test, Cochran-Armitage trend test, maximin efficiency robust test, MAX3 test and genetic model selection test. By Yong Zang, Wingkam Fung and Gang Zheng.

RcmdrPlugin.DoE **Rcmdr** plugin for (industrial) Design of Experiments. Currently in beta status. By Ulrike Grömping.

RcmdrPlugin.qual An **Rcmdr** plug-in based on the Quality control class Stat 4300. By Erin Hodgess.

ReacTran Routines for developing models that describe reaction and advective-diffusive transport in one, two or three dimensions. Includes transport routines in porous media, in estuaries, and in bodies with variable shape. By Karline Soetaert and Filip Meysman.

ReadImages Functions for reading JPEG and PNG files, requiring libjpeg. By Markus Loecher.

RelativeRisk Relative risk estimation for prospective and retrospective data. By Bob Wheeler.

Rmpfr R MPFR - Multiple Precision Floating-Point Reliable. Aims to provide S4 classes and methods for arithmetic including transcendental ("special") functions for arbitrary precision floating point numbers. To this end, it interfaces to the LGPL'ed MPFR (Multiple Precision Floating-Point Reliable) Library which itself is based on the GMP (GNU Multiple Precision) Library. By Martin Mächler.

Rniftilib R interface to nifticlib to read/write ANALYZE(TM)7.5/NIfTI-1 volume images. By Oliver Granert.

RthroughExcelWorkbooksInstaller Workbooks in Excel that illustrate statistical concepts by accessing R functions from Excel. These workbooks use the automatic recalculation mode of Excel to update calculations and graphs in R. Downloads an executable which installs the workbooks on MS Windows systems where **RExcel** has already been installed. By Richard M. Heiberger and Erich Neuwirth.

SAFD Statistical Analysis of Fuzzy Data. Aims to provide some basic functions for doing statistics with one dimensional fuzzy data (in the form of polygonal fuzzy numbers). Contains functions for the basic operations on the class of fuzzy numbers (sum, scalar product, mean, Hukuhara difference) as well as for calculating (Bertoluzza) distance, sample variance, sample covariance, sample correlation, and the Dempster-Shafer (levelwise) histogram. Includes functionality to simulate fuzzy random variables, bootstrap tests for the equality of means, and to do linear regression given trapezoidal fuzzy data. By Wolfgang Trutschnig and Asun Lubiano.

SEL Semiparametric elicitation. Implements a novel method for transferring expert statements about an uncertain bounded quantity into a probability distribution (see Bornkamp and Ickstadt (2009) for a detailed description). For this purpose B-splines are used, and the density is obtained by penalized least squares, where the penalty encourages to distribute probability mass as uniformly as possible. Provides methods for fitting the expert's distribution as well as methods for evaluating the underlying density and cdf. In addition methods for plotting the expert's distribution, drawing random numbers and calculating quantiles of the expert's distribution are provided. By Björn Bornkamp.

SHARE SNP-Haplotype Adaptive REgression (SHARE): an adaptive algorithm to select the most informative set of SNPs for genetic association. By James Y. Dai.

SIS (Iterative) Sure Independence Screening for Generalized Linear Models and Cox's Proportional Hazards models. By Jianqing Fan, Yang Feng, Richard Samworth and Yichao Wu.

SWordInstaller SWord: Use R in Microsoft Word (Installer). Creating articles and reports in Word is easy. Adding R results (figures, tables, summaries) requires manually copying the data from R to Word. Changing the data requires redoing the analyses in R and re-inserting the results into Word manually. SWord integrates R scripts and results into Word documents. Documents can be edited and read even without R installed (or without any knowledge of R). The functionality of embedding the scripts is somewhat similar to what Sweave does for LaTeX documents. By Thomas Baier.

SigWinR Implementation of the SigWin-detector for the detection of regions of increased or decreased gene expression (RIDGEs and anti-RIDGES) in transcriptome maps and the presentation in so called RIDGEOGRAMs as described by Marcia A. Inda et al. (2008). By Wim de Leeuw.

TripleR Social Relation Model (SRM) analyses for single round-robin groups. These analyses are either based on one manifest variable, one latent construct measured by two manifest variables, two manifest variables and their bivariate relations, or two latent constructs each measured by two manifest variables (in the last

case, four Round-Robin matrices have to be provided). By S.C. Schmukle, F.D. Schoenbrodt and M.D. Back.

WMCapacity A GUI implementation of hierarchical Bayesian models of working memory, used for analyzing change detection data. By Richard D. Morey.

YieldCurve Modeling and estimation of the yield curve, implementing the Nelson-Siegel, Diebold-Li and Svensson models. Also includes the data of the term structure of interest rate of Federal Reserve and European Central Bank. By Sergio Salvino Guirreri.

aCGH.Spline Robust spline interpolation for dual color array comparative genomic hybridisation data. By Tomas William Fitzgerald.

adaptTest Adaptive two-stage tests. Currently, four tests are included: Bauer and Koehne (1994), Lehmacher and Wassmer (1999), Vandemeulebroecke (2006), and the horizontal conditional error function. By Marc Vandemeulebroecke.

amer Fit generalized additive mixed models based on the mixed model algorithm of **lme4**. By Fabian Scheipl.

anchors Statistical analysis of surveys with anchoring vignettes. By Jonathan Wand, Gary King, and Olivia Lau.

ant Version of the ant apache build tool, with a few R specific tasks to ease use of ant within an R package. By Romain François.

aroma.affymetrix Analysis of large Affymetrix microarray data sets. Implements classes for files and sets of files for various Affymetrix file formats, e.g., AffymetrixCdfFile, AffymetrixCelFile, and AffymetrixCelSet. These are designed to be memory efficient but still being fast. The idea is to keep all data on file and only read data into memory when needed. Clever caching mechanisms are used to minimize the overhead of data I/O. All of the above is hidden in the package API and for the developer (and the end user), the data is queried as if it lives in memory. With this design it is only the diskspace that limits what number of arrays can be analyzed. By Henrik Bengtsson, Ken Simpson, Elizabeth Purdom, and Mark Robinson.

aroma.core Private support package for **aroma.affymetrix** et al., with the package API still in alpha/beta. By Henrik Bengtsson.

atm Creation of additive models with semiparametric predictors, emphasizing term objects, especially (1) implementation of a term class hierarchy, and (2) interpretation and evaluation of term estimates as functions of explanatories. By Charlotte Maia.

atmi Analysis and usage of the trading rules, which are based on technical market indicators as well as on the time series analysis. By Waldemar Kemler and Peter Schaffner.

bclust Bayesian clustering using spike-and-slab hierarchical model, suitable for clustering high-dimensional data. Builds a dendrogram with log posterior as a natural distance defined by the model. Can also compute Bayesian discrimination probabilities equivalent to the implemented Bayesian clustering. Spike-and-Slab models are adopted in a way to be able to produce an importance measure for clustering and discriminant variables. The method works properly for data with small sample size and high dimensions. The model parameter estimation maybe difficult, depending on data structure and the chosen distribution family. By Vahid Partovi Nia and Anthony C. Davison.

bcv Cross-Validation for the SVD (Bi-Cross-Validation). This package implements methods for choosing the rank of an SVD approximation via cross validation. It provides both Gabriel-style "block" holdouts and Wold-style "speckled" holdouts. Also included is an implementation of the SVDImpute algorithm. See Owen & Perry's 2009 AOAS article (http://arxiv.org/abs/0908.2062) and Perry's 2009 PhD thesis (http://arxiv.org/abs/0909.3052) for more information about Bi-crossvalidation. By Patrick O. Perry.

bdoc Bayesian Discrete Ordered Classification of DNA Barcodes. By Michael Anderson.

bdsmatrix Routines for Block Diagonal Symmetric matrices, a special case of sparse matrices, used by **coxme** and **kinship**. By Terry Therneau.

bestglm Best subset GLM using AIC, BIC, EBIC, BICq or Cross-Validation. For the normal case, the "leaps" is used; otherwise, a slower exhaustive search. By A.I. McLeod and Changjiang Xu.

binomSamSize Confidence intervals and sample size determination for a binomial proportion under simple random sampling and pooled sampling. Such computations are e.g. of interest when investigating the incidence or prevalence in populations. Contains functions to compute coverage probabilities and coverage coefficients of the provided confidence intervals procedures. Sample size calculations are based on expected length. By Michael Höhle with contributions by Wei Liu.

bootRes Calculation of bootstrapped response and correlation functions for use in dendroclimatology. By Christian Zang.

bvpSolve Solvers for boundary value problems (BVPs) of systems of ordinary differential equations (ODEs) via an interface to the FORTRAN function `twpbvp` and an R implementation of the shooting method. By Karline Soetaert.

caGUI A Tcl/Tk GUI for computation and visualization of simple, multiple and joint correspondence analysis with the **ca** package. By Angelos Markos.

caroline A collection of functions useful for various general situations. By David M. Schruth.

cimis A set of functions for retrieving data from CIMIS, the California Irrigation Management Information System. By Greg Hirson.

cmm Categorical Marginal Models: Quite extensive package for the estimation of marginal models for categorical data. By Wicher Bergsma and Andries van der Ark.

cmprskContin Estimation and testing of continuous mark-specific relative risks in two groups as described in Gilbert, McKeague & Sun (2008). Implements the methods presented in the paper for testing mark-specific hazards ratios and for estimation of mark-specific incidence ratios that are cumulative in time or cumulative in both time and the continuous mark. By Peter Gilbert, Ian McKeague, and Yanqing Sun.

cmrutils A collection of useful helper routines developed by students of the Center for the Mathematical Research, Stankin, Moscow. By Andrey Paramonov.

colbycol Read big text files column by column: tries to solve the memory restrictions posed by such files by breaking them into columns which are subsequently read individually into R. By Carlos J. Gil Bellosta.

condGEE Solves for the mean parameters, the variance parameter, and their asymptotic variance in a conditional GEE for recurrent event gap times, as described by Clement and Strawderman (2009). Makes a parametric assumption for the length of the censored gap time. By David Clement.

countrycode Create and manipulate data frames that contain country names or country coding schemes. Standardizes country names, converts them into one of seven coding schemes, assigns region descriptors, and generates empty dyadic or country-year dataframes from the coding schemes. By Vincent Arel-Bundock.

coxme Mixed Effects Cox Models: Cox proportional hazards models containing Gaussian random effects, also known as frailty models. By Terry Therneau.

crantastic Various R tools for `http://crantastic.org/`. By Bjørn Arild Maeland.

cshapes Package for CShapes, a GIS dataset of country boundaries (1946–2008). Includes functions for data extraction and the computation of weights matrices. By Nils B. Weidmann, Doreen Kuse, and Kristian Skrede Gleditsch.

cudaBayesreg CUDA Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis. Compute Unified Device Architecture (CUDA) is a software platform for massively parallel high-performance computing on NVIDIA GPUs. This package provides a CUDA implementation of a Bayesian multilevel model for the analysis of brain fMRI data. By Adelino Ferreira da Silva.

dcemri A collection of routines and documentation that allows one to perform a quantitative analysis of dynamic contrast-enhanced or diffusion-weighted MRI data. Medical imaging data should be organized using either the Analyze or NIfTI data formats. By Brandon Whitcher and Volker Schmid, with contributions from Andrew Thornton.

descr Descriptive statistics: functions to describe weighted categorical variables and functions to facilitate the character encoding conversion of objects. By Jakson Aquino. Includes R source code and/or documentation written by Dirk Enzmann, Marc Schwartz, and Nitin Jain.

desire Harrington and Derringer-Suich type desirability functions. By Heike Trautmann and Detlef Steuer and Olaf Mersmann.

difR Some traditional methods to detect dichotomous differential item functioning (DIF) in psychometrics. Both uniform and non-uniform DIF effects can be detected, with methods relying upon item response models or not. Some methods deal with more than one focal group. By Sébastien Béland, David Magis and Gilles Raîche.

digeR An easy to use Graphical User Interface for spots correlation analysis, score plot, classification, feature selection and power analysis for 2D DIGE experiment data. By Yue Fan, Thomas Brendan Murphy, and R. William G. Watson.

dirmult Estimate parameters in Dirichlet-Multinomial and compute profile log-likelihoods. By Torben Tvedebrink.

dlnm Distributed Lag Non-linear Models. Contains functions to specify basis and cross-basis matrices in order to run distributed lag models and their non-linear extension, then to predict and graph the results for a fitted model. By Antonio Gasparrini and Ben Armstrong.

doMC Provides a parallel backend for the **foreach** `%dopar%` function using Simon Urbanek's **multicore** package. By REvolution Computing.

doSNOW Provides a parallel backend for the **foreach** `%dopar%` function using Luke Tierney's **snow** package. By REvolution Computing.

dummies Expands factors, characters and other eligible classes into dummy/indicator variables. By Christopher Brown.

dynGraph Interactive visualization of dataframes and factorial planes. By Julien Durand, Sébastien Lê.

el.convex Empirical likelihood ratio tests for means. By Dan Yang, Dylan Small.

endogMNP Fits a Bayesian multinomial probit model with endogenous selection, which is sometimes called an endogenous switching model. This can be used to model discrete choice data when respondents select themselves into one of several groups. This package is based on the **MNP** package by Kosuke Imai and David A. van Dyk. By Lane F. Burgette.

estout Estimates Output: stores the estimates of several models, and formats these to a table of the form estimate starred and standard error. below. Default output is LaTeX but output to CSV for later editing in a spreadsheet tool is possible as well. Works for linear models and panel models from package **plm**. By Felix Kaminsky.

exact2x2 Exact Conditional Tests and Confidence Intervals for 2×2 tables. Calculates Fisher's exact and Blaker's exact tests. By M.P. Fay.

farmR A mixed integer description of an arable farm. Finds optimal farming plans given economic and social preference information. By Ira R. Cooke.

fds Functional data sets. By Han Lin Shang and Rob J Hyndman.

flsa Path algorithm for the general Fused Lasso Signal Approximator. By Holger Höfling.

flubase Estimates the baseline of mortality free of influenza epidemics, and the respective excess deaths, for more than one time series (age groups, gender, regions, etc.). By Nunes B, Natario I and Carvalho L.

foreach Support for the foreach looping construct. Foreach is an idiom that allows for iterating over elements in a collection, without the use of an explicit loop counter. Intended to be used for its return value, rather than for its side effects. Using **foreach** without side effects also facilitates executing the loop in parallel. By REvolution Computing.

fractalrock Generate fractal time series with non-normal returns distribution. The basic principle driving fractal generation of time series is that data is generated iteratively based on increasing levels of resolution. The initial series is defined by a so-called initiator pattern and then generators are used to replace each segment of the initial pattern. Regular, repeatable patterns can be produced by using the same seed and generators. By using a set of generators, non-repeatable time series can be produced. This technique is the basis of the fractal time series process in this package. By Brian Lee Yung Rowe.

frbf Implementation of the "Flexible kernels for RBF network" algorithm. By Fernando Martins.

freqMAP Estimates a frequency Moving Average Plot (MAP) from multinomial data and a continuous covariate. The frequency MAP is a moving average estimate of category frequencies, where frequency means and posterior bounds are estimated. Comparisons of two frequency MAPs as well as odds ratios can be plotted. By Colin McCulloch.

ftsa Functional time series analysis. By Rob J Hyndman and Han Lin Shang.

gPdtest Computes the bootstrap goodness-of-fit test for the generalized Pareto distribution proposed by Villaseñor-Alva and González-Estrada (2009). The null hypothesis includes heavy and non-heavy tailded gPd's. Also provides functionality for fitting the gPd to data using the parameter estimation methods proposed in the same article. By Elizabeth González Estrada, José A. Villaseñor Alva.

gRapHD Efficient selection of undirected graphical models for high-dimensional datasets. Provides tools for selecting trees, forests and decomposable models minimizing information criteria such as AIC or BIC, and for displaying the independence graphs of the models. Also some useful tools for analyzing graphical structures. It supports the use of discrete, continuous, or both types of variables. By Gabriel Coelho Goncalves de Abreu, Rodrigo Labouriau, and David Edwards.

gamesNws Playing games using a NWS Server. Allows playing different card games (e.g. uno, poker, ...) using an NWS Server as the card table. By Markus Schmidberger and Fabian Grandke.

gamlss.data Data for GAMLSS models. By Mikis Stasinopoulos and Bob Rigby.

gamm4 Fit generalized additive mixed models via a version of **mgcv**'s `gamm()` function, using **lme4** for estimation via Fabian Scheipl's trick. By Simon Wood.

gausspred Predict the discrete response based on selected high dimensional features, such as gene expression data. The data are modeled with Bayesian Gaussian models. When a large number of features are available, one may like to select only a subset of features to use, typically those features strongly correlated with the response in training cases. Such a feature selection procedure is however invalid since the relationship between the response and the features will appear stronger. This package provides a way to avoid this bias and yields well-calibrated prediction for the test cases when one uses F-statistic to select features. By Longhai Li.

geneARMA Simulate, model, and display data from a time-course microarray experiment with periodic gene expression. Fits models in a normal mixture model framework with mean approximated by a truncated Fourier series and covariance structure modeled by an ARMA(p,q) process. Estimation is performed with the EM algorithm. By Timothy McMurry and Arthur Berg.

geneListPie Map a gene list to function categories defined in GOSlim or Kegg. The results can be plotted as a pie chart to provide a quick view of the distribution of the gene list among the function categories. By Xutao Deng.

glmdm Simulation of GLMDM. By Jeff Gill, George Casella, Minjung Kyung and Jonathan Rapkin.

glmperm A permutation test for inference in generalized linear models. Useful when parameter estimates in ordinary GLMs fail to converge or are unreliable due to small sample size. By Wiebke Werft and Douglas M. Potter.

glmulti GLM model selection and multimodel inference made easy: automated model selection for GLMs. Provides a wrapper for `glm()` and similar functions, automatically generating all possible models (under constraints set by the user) with the specified response and explanatory variables, and finding the best models in terms of some Information Criterion (AIC, AICc or BIC). Can handle very large numbers of candidate models. Features a Genetic Algorithm to find the best models when an exhaustive screening of the candidates is not feasible. By Vincent Calcagno.

gnumeric Read data from files readable by gnumeric. Can read a whole sheet or a range, from several file formats, including the native format of gnumeric. Reading is done by using ssconvert (a file converter utility included in the gnumeric distribution `http://projects.gnome.org/gnumeric/`) to convert the requested part to CSV. By Karoly Antal.

gsDesign Derive group sequential designs and describe their properties. By Keaven Anderson.

iGenomicViewer Tool for sending interactive bioinformatic heatmaps with tool-tip content. By Daniel P Gaile, Lori A. Shepherd, Lara Sucheston, Andrew Bruno, and Kenneth F. Manly.

infotheo Implements various measures of information theory based on several entropy estimators. By Patrick E. Meyer.

inlinedocs Convert inline comments to documentation. Generates Rd files from R source code with comments, providing for quick, sustainable package development. The syntax keeps code and documentation close together, and is inspired by the Don't Repeat Yourself principle. By Toby Dylan Hocking.

intamap Classes and methods for automated spatial interpolation. By Edzer Pebesma, Jon Skøien and others.

integrOmics Integrate Omics data project. Supplies two efficient methodologies, regularized CCA and sparse PLS, to unravel relationships between two heterogeneous data sets of size $n \times p$ and $n \times q$ where the p and q variables are measured on the same samples or individuals n. These data may come from high throughput technologies, such as omics data (e.g., transcriptomics, metabolomics or proteomics data) that require an integrative or joint analysis. However, **integrOmics** can also be applied to any other large data sets where $p + q \gg n$. rCCA is a regularized version of CCA to deal with the large number of variables. sPLS allows variable selection in a one step procedure and two frameworks are proposed: regression and canonical analysis. Numerous graphical outputs are provided to help interpreting the results. By Sébastien Dejean, Ignacio González and Kim-Anh Lê Cao.

integrativeME integrative mixture of experts. Mixture of experts models (Jacobs et al., 1991) were introduced to account for nonlinearities and other complexities in the data, and are of interest due to their wide applicability and the advantages of fast learning via the expectation-maximization (EM) algorithm. The package features an ME extension to combine categorical clinical factors and continuous microarray data in a binary classification framework to analyze cancer studies. By Kim-Anh Lê Cao.

interval Weighted Logrank Tests and NPMLE for interval censored data. By Michael P. Fay.

isa2 The Iterative Signature Algorithm (ISA), a biclustering algorithm that finds modules in an input matrix. A module or bicluster is a block of the reordered input matrix. By Gábor Csárdi.

iterators Iterator construct for R. Iterators allow a programmer to traverse through all the elements of a vector, list, or other collection of data. By REvolution Computing.

kmi Kaplan-Meier multiple imputation to recover the missing potential censoring information from competing risks events, so that standard right-censored methods could be applied to the imputed data sets to perform analyses of the cumulative incidence functions. By Arthur Allignol.

latdiag Writes a file of commands for the dot program to draw a graph proposed by Rosenbaum and useful for checking some properties of various sorts of latent scale. By Michael Dewey.

latticedl Direct labeling functions that use the **lattice** package. By Toby Dylan Hocking.

lda Collapsed Gibbs sampling methods for topic models. Implements Latent Dirichlet allocation (LDA) and related models, including (but not limited to) sLDA, corrLDA, and the mixed-membership stochastic blockmodel. Inference for all of these models is implemented via a fast collapsed Gibbs sampler written in C. Utility functions for reading/writing data typically used in topic models, as well as tools for examining posterior distributions are also included. By Jonathan Chang.

lemma Laplace approximated EM Microarray Analysis. LEMMA is used to detect "nonnull genes" — genes for which the average response in treatment group 1 is significantly different from the average response in group 2, in normalized microarray data. LEMMA is an implementation of an approximate EM algorithm to estimate the parameters in the assumed linear model in Bar, Booth, Schifano, Wells (2009). By Haim Bar and Elizabeth Schifano.

longitudinalData Tools for Longitudinal Data. By Christophe Genolini.

lordif LOgistic Regression Differential Item Functioning (DIF) using Item Response Theory (IRT): analysis of DIF for dichotomous and polytomous items using an iterative hybrid of (ordinal) logistic regression and IRT. By Seung W. Choi, with contributions from Laura E. Gibbons and Paul K. Crane.

mar1s Multiplicative AR(1) with Seasonal Processes, a stochastic process model built on top of AR(1). Provides the following procedures for MAR(1)S processes: fit, compose, decompose, advanced simulate and predict. By Andrey Paramonov.

marelacTeaching Datasets and tutorials for use in the MArine, Riverine, Estuarine, LAcustrine and Coastal sciences. By Karline Soetaert, Thomas Petzoldt, and Filip Meysman.

maticce MApping Transitions In Continuous Character Evolution. Tools for an information-theoretic approach to estimating the probability of continuous character transitions on phylogenetic trees. By Andrew Hipp, with contributions from Marcial Escudero.

matrixStats Methods that apply to rows and columns of a matrix. Methods are optimized for speed and memory. Currently in a beta phase. By Henrik Bengtsson (partly by Robert Gentleman).

mcclust Methods for processing a sample of (hard) clusterings, e.g., the MCMC output of a Bayesian clustering model. Includes methods that find a single best clustering to represent the sample, which are based on the posterior similarity matrix or a relabeling algorithm. By Arno Fritsch.

mediation Parametric and nonparametric causal mediation analysis. Implements the methods and suggestions in Imai, Keele, and Yamamoto (2008) and Imai, Keele, Tingley (2009). In addition to the estimation of causal mediation effects, allows researchers to conduct sensitivity analysis for certain parametric models. By Luke Keele, Dustin Tingley, Teppei Yamamoto, Kosuke Imai.

metafor A collection of functions for conducting meta-analyses in R. Fixed- and random-effects models (with and without moderators) can be fitted via the general linear (mixed-effects) model. For 2×2 table data, the Mantel-Haenszel and Peto's method are also implemented. By Wolfgang Viechtbauer.

mixfdr Fits normal mixture models to data and uses them to compute effect size estimates and local and tail area false discovery rates. To make this precise, suppose you have many normally distributed zs, and each z_i has mean δ_i. This package will estimate δ_i based on the zs (effect sizes), $P(\delta_i = 0|z_i)$ (local false discovery rates) and $P(\delta_i = 0 \mid |z_i| > z)$ (tail area false discovery rates). By Omkar Muralidharan, with many suggestions from Bradley Efron.

modTempEff Modeling temperature effects using time series data. Fits a constrained segmented distributed lag regression model to epidemiological time series of mortality, temperature, and other confounders. By Vito M. R. Muggeo.

mrt Datasets and functions from Wright and London's Modern Regression Techniques. By Daniel B. Wright.

mstate Functions for data preparation, descriptives, hazard estimation and prediction with Aalen-Johansen or simulation in competing risks and multi-state models. By Hein Putter, Liesbeth de Wreede, and Marta Fiocco.

muRL Mailmerge using R, LaTeX, and the Web. Provides mailmerge methods for reading spreadsheets of addresses and other relevant information to create standardized but customizable letters. Provides a method for mapping US ZIP codes, including those of letter recipients. Provides a method for parsing and processing HTML code from online job postings of the American Political Science Association. By Ryan T. Moore and Andrew Reeves.

multmod Testing of multiple outcomes using i.i.d. decompositions. By Christian B. Pipper, Christian Ritz.

munsell Functions for exploring and using the Munsell colour system. By Charlotte Wickham.

mvShapiroTest Generalized Shapiro-Wilk test for multivariate normality as proposed by Villaseñor-Alva and González-Estrada (2009). By Elizabeth González Estrada and José A. Villaseñor Alva.

mvsf Multivariate generalization of the Shapiro-Francia test for normality. By David Delmail.

nbpMatching Functions for non-bipartite optimal matching. By Bo Lu, Robert Greevy, Cole Beck.

nnclust Nearest-neighbor tools for clustering. Finds nearest neighbors and the minimum spanning tree (MST) for large data sets, and does clustering using the MST. By Thomas Lumley.

nodeHarvest Compute and visualize the Node Harvest estimator. Node harvest is a simple interpretable tree-like estimator for high-dimensional regression and classification. A few nodes are selected from an initially large ensemble of nodes, each associated with a positive weight. New observations can fall into one or several nodes and predictions are the weighted average response across all these groups. By Nicolai Meinshausen.

nparLD Nonparametric Analysis of Longitudinal Data in Factorial Experiments. By Kimihiro Noguchi, Mahbub Latif, Karthinathan Thangavelu, Frank Konietschke, Yulia R. Gel, and Edgar Brunner.

oblique.tree Grows oblique trees to classification data. By Alfred Truong.

odfWeave.survey odfWeave support for the **survey** package. By Thomas Lumley.

oosp Object Oriented Statistical Programming. Support for OOSP, especially by extending S3 capabilities, providing pointer and component objects, and providing basic support for symbolic-numeric statistical programming. By Charlotte Maia.

optBiomarker Estimation of optimal number of biomarkers for two-group microarray based classifications at a given error tolerance level for various classification rules. By Mizanur Khondoker.

p3state.msm Analyzing survival data from illness-death models. By Luís Meira-Machado and Javier Roca-Pardiñas.

packS4 Toy example of an S4 package, to illustrate the book "Petit Manuel de Programmation Orientee Objet sous R". By Christophe Genolini.

packdep Elucidates the dependencies between user-contributed R packages and identifies key packages according to social network analysis metrics. By Radhakrishnan Nagarajan and Marco Scutari.

pamm Simulation functions to assess or explore the power of a dataset to estimate significant random effects (intercept or slope) in a mixed model. Based on the **lme4** package. By Julien Martin.

parser Detailed source code parser, based on the standard R parser but organizing the information differently. By Romain François.

pedigreemm Fit pedigree-based mixed-effects models. By Douglas Bates and Ana Ines Vazquez.

pegas Population and Evolutionary Genetics Analysis System. Provides functions for reading, writing, plotting, analysing, and manipulating allelic data, and for the analysis of population nucleotide sequences including coalescence analyses. By Emmanuel Paradis.

pendensity Estimation of penalized (conditional) densities. By Christian Schellhase.

perm Exact or Asymptotic permutation tests. By Michael Fay.

pgs Precision of Geometric Sampling. Computes mean squared errors of stereological predictors. By Kien Kieu and Marianne Mora.

phull Computes the p-hull of a finite planar set, which is a generalization of the convex hull, X-Y hull and bounding rectangle. A fast, $O(n \log n)$ Graham-scan based routine is used. By Marek Gagolewski.

phybase Basic functions for phylogenetic analysis. Provides functions to read, write, manipulate, estimate, and summarize phylogenetic trees including species trees which contain not only the topology and branch lengths but also population sizes. The input/output functions can read tree files in which trees are presented in parenthetic format. The trees are read in as a string and then transformed to a matrix which describes the relationship of nodes and branch lengths. The nodes matrix provides an easy access for developers to further manipulate the tree, while the tree string provides interface with other phylogenetic R packages such as **ape**. The input/output functions can also be used to change the format of tree files between NEXUS and PHYLIP. Some basic functions have already been established in the package for manipulating trees such as deleting and swapping nodes, rooting and unrooting trees, changing the root of the tree. The package also includes functions for summarizing phylogenetic trees, calculating the coalescence time, population size, and tree distance, and to estimate the species tree from multiple gene trees. By Liang Liu.

plus Penalized Linear Unbiased Selection. Efficient procedures for fitting an entire regression sequences with different model types. By Cun-Hui Zhang and Ofer Melnik.

pooh Partial Orders and Relations: functions for computing closures of relations. By Charles J. Geyer.

potts Markov Chain Monte Carlo for Potts Models. By Charles J. Geyer.

psgp Projected Spatial Gaussian Process methods for package **intamap**. By Ben Ingram and Remi Barillec.

rWMBAT The William and Mary Bayesian Analysis Tool. By Karl Kuschner, Qian Si and William Cooke, College of William and Mary.

rainbow Rainbow plots, bagplots and boxplots for functional data. By Han Lin Shang and Rob J Hyndman.

rankhazard Rank-hazard plots (Karvanen and Harrell, Statistics in Medicine, 2009) which visualize the relative importance of covariates in a proportional hazards model. The key idea is to rank the covariate values and plot the relative hazard as a function of ranks scaled to interval $[0,1]$. The relative hazard is plotted with respect to the reference hazard, which can be e.g. the hazard related to the median of the covariate. Transformation to scaled ranks allows plotting of covariates measured in different units in the same graph, which helps in the interpretation of the epidemiological relevance of the covariates. Rank-hazard plots show the difference of hazards between the extremes of the covariate values present in the data and can be used as a tool to check if the proportional hazards assumption leads to reasonable estimates for individuals with extreme covariate values. Alternative covariate definitions or different transformations applied to covariates can be also compared using rank-hazard plots. By Juha Karvanen.

ringscale Implementation of the "Ringscale" method as proposed in the student research project "Detection of faint companions around young stars in speckle patterns of VLT/NACO cube mode images by means of post-processing" at the Friedrich-Schiller-University of Jena. By Daniel Haase.

rioja Analysis of Quaternary science data, including constrained clustering, WA, WAPLS, IKFA, MLRC and MAT transfer functions, and stratigraphic diagrams. By Steve Juggins.

ripa R Image Processing and Analysis. Makes it possible to process and analyze RGB, LAN (multispectral) and AVIRIS (hyperspectral) images. By Talita Perciano, with contributions from Alejandro C Frery.

rms Regression Modeling Strategies: regression modeling, testing, estimation, validation, graphics, prediction, and typesetting by storing enhanced model design attributes in the fit. A collection of about 225 functions that assist with and streamline modeling. Also contains functions for binary and ordinal logistic

regression models and the Buckley-James multiple regression model for right-censored responses, and implements penalized maximum likelihood estimation for logistic and ordinary linear models. **rms** works with almost any regression model, but it was especially written to work with binary or ordinal logistic regression, Cox regression, accelerated failure time models, ordinary linear models, the Buckley-James model, generalized least squares for serially or spatially correlated observations, generalized linear models, and quantile regression. By Frank E Harrell Jr.

robustX eXperimental eXtraneous eXtraordinary functionality for Robust Statistics. By Werner Stahel, Martin Mächler and potentially others.

rpartOrdinal Functions that can be called in conjunction with **rpart** for deriving a classification tree when the response to be predicted is ordinal. By Kellie J. Archer.

rrv Random Return Variables. Currently provides limited support for formatting money. By Charlotte Maia.

sBF Smooth Backfitting for additive models using Nadaraya-Watson estimator. By A. Arcagni, L. Bagnato.

safeBinaryRegression Safe Binary Regression. Overloads the `glm()` function in the **stats** package so that a test for the existence of the maximum likelihood estimate is included in the fitting procedure for binary regression models. By Kjell Konis.

sculpt3d A simple toolbar GUI for brushing RGL plots. Controls for simple brushing, highlighting, labeling, and mouseMode changes are provided by point-and-click rather than through the R terminal interface. By Justin Donaldson.

sddpack Semidiscrete Decomposition (SDD), which approximates a matrix as a weighted sum of outer products formed by vectors with entries constrained to be in the set $\{-1,0,1\}$. By Tamara G. Kolda and Dianne P. O'Leary.

sdef Synthesizing list of Differentially Expressed Features. Performs two tests to evaluate if experiments are associated and returns a list of interesting features common to all the experiments. By Alberto Cassese and Marta Blangiardo.

season Routines for the seasonal analysis of health data, including regression models, time-stratified case-crossover, plotting functions and residual checks. By Adrian Barnett and Peter Baker.

sendmailR A simple SMTP client which provides a portable solution for sending emails from within R. By Olaf Mersmann.

simFrame A general framework for statistical simulation. By Andreas Alfons.

simctest Sequential (or Safe) Implementation of Monte Carlo tests with uniformly bounded resampling risk. Features efficient computation of p-values for Monte Carlo tests, e.g., bootstrap tests. By Axel Gandy.

skellam Functions for the Skellam distribution, including: pmf, cdf, quantiles and random variates. By Jerry W. Lewis.

skmeans Algorithms to compute spherical k-means partitions. Features several methods, including a genetic and a simple fixed-point algorithm and an interface to the CLUTO vcluster program. By Kurt Hornik, Ingo Feinerer and Martin Kober.

slam Sparse Lightweight Arrays and Matrices. Data structures and algorithms for sparse arrays and matrices, based on index arrays and simple triplet representations, respectively. By Kurt Hornik, David Meyer, and Christian Buchta.

sos R related Search Engines. By Spencer Graves, Sundar Dorai-Raj, and Romain François.

spatcounts Fit spatial CAR count regression models using MCMC. By Holger Schabenberger.

speedglm Fitting LMs and GLMs to large data sets by updating algorithms. By Marco ENEA.

speff2trial Semiparametric efficient estimation for a two-sample treatment effect: performs estimation and testing of the treatment effect in a 2-group randomized clinical trial with a quantitative or dichotomous endpoint. The method is a special case of Robins, Rotnitzky, and Zhao (1994, JASA). It improves efficiency by leveraging baseline predictors of the endpoint. The method uses inverse probability weighting to provide unbiased estimation when the endpoint is missing at random. By Michal Juraska, with contributions from Peter B. Gilbert, Min Zhang, Marie Davidian, Anastasios A. Tsiatis and Xiaomin Lu.

stringkernels Gapped and word-based string kernels for use with **kernlab**. By Martin Kober.

sublogo Visualize correlation in biological sequence data using sublogo dendrogram plots. By Toby Dylan Hocking.

sugaR Plots to help optimizing diabetes therapy. Provides a series of plots to integrate glucose levels, basal rate, activities, events and carbohydrate uptake on a single page in a humanely interpretable manner. It is meant for best-possibly representing the content of a well-curated diabetes diary of up to a week's time or of up to seven comparable days, from which conclusions for adjusting the individual treatment shall be drawn. By Steffen Möller.

svDialogs SciViews GUI API: dialog boxes. Rapidly construct dialog boxes for your GUI, including an automatic function assistant. By Philippe Grosjean.

svSweave SciViews GUI API: Sweave support functions. By Philippe Grosjean.

svTools SciViews GUI API: tools, aimed at wrapping some of the functionalities of the packages **tools**, **utils** and **codetools** into a nicer format so that an IDE can use them. By Romain François.

svUnit SciViews GUI API: unit testing. A complete unit test system and functions to implement its GUI part. By Philippe Grosjean.

svWidgets SciViews GUI API: widgets & windows. High level management of widgets, windows and other graphical resources. By Philippe Grosjean.

textcat N-Gram based text categorization. By Kurt Hornik, Johannes Rauch, Christian Buchta, and Ingo Feinerer.

tikzDevice A device for R graphics output in PGF/TikZ format. Enables LaTeX-ready output from R graphics functions, with LaTeX mathematics that can be typeset directly into labels and annotations. Graphics produced this way can also be annotated with custom TikZ commands. By Charlie Sharpsteen and Cameron Bracken.

tm.plugin.mail A plug-in for the **tm** text mining framework providing mail handling functionality. By Ingo Feinerer.

tnet Analysis of Weighted, Two-mode, and Longitudinal networks. By Tore Opsahl.

tolerance Functions for calculating tolerance intervals. Tolerance limits provide the limits between which we can expect to find a specified proportion of a population with a given level of confidence. Provides functions for estimating tolerance limits for various distributions, and plotting tolerance limits of continuous random variables. By Derek S. Young.

tractor.base Basic functions for the TractoR (tractography with R) bundle. Consists of functions for working with magnetic resonance images. Can read and write image files stored in Analyze, NIfTI and DICOM formats (DICOM support is read only), generate images for use as regions of interest, and manipulate and visualize images. By Jon Clayden.

treethresh Methods for Tree-based Local Adaptive Thresholding. By Ludger Evers and Tim Heaton.

ttrTests Standard Backtests for Technical Trading Rules in Financial Data. Four core functions evaluate the efficacy of a technical trading rule: conditional return statistics, bootstrap resampling statistics, test for data snooping bias among parameter choices, and robustness of parameter choices. By David St John.

ttutils Some utility functions. By Thorn Thaler.

twitteR An R based Twitter client via an interface to the Twitter web API. By Jeff Gentry.

vegetarian Jost Diversity Measures for Community Data. Computes diversity for community data sets using the methods outlined by Jost (2006, 2007), which offer the advantage of providing diversity numbers equivalents, independent alpha and beta diversities, and the ability to incorporate "order" as a continuous measure of the importance of rare species in the metrics. Computes alpha diversities, beta diversities, gamma diversities, and similarity indices. Confidence intervals for diversity measures are calculated using a bootstrap method described by Chao et al. (2008). By Noah Charney, Sydne Record.

yhat Methods for variance partitioning for linear models and canonical correlation and methods for interpreting regression effects using beta weights, standardized beta weights, structure coefficients, and adjusted effect sizes. By Kim Nimon and J. Kyle Roberts.

Other changes

- Recommended bundle **VR** was unbundled into the recommended packages **MASS**, **class**, **nnet** and **spatial**, and moved to the Archive.
- Bundle **BACCO** was unbundled into its packages **approximator**, **calibrator** and **emulator**, and moved to the Archive.
- Bundle **forecasting** was unbundled into its packages **Mcomp**, **expsmooth**, **fma** and **forecast**, and moved to the Archive.

- Bundle **hoa** was unbundled into its packages **cond**, **csampling**, **marg** and **nlreg**, and moved to the Archive.
- Packages **ARES**, **MDD**, **GeneNT**, **HTMLapplets**, **RiboSort**, **SLmisc**, **UNF**, **WaveCGH**, **WeedMap**, **agreement**, **bivpois**, **boost**, **clac**, **crq**, **ggplot**, **mclust02**, **norm**, **partsm**, **pwr**, **sma**, **supclust**, **uroot** and **verify** were moved to the Archive.
- Packages **Multiclasstesting**, **Rlab**, **intcox**, **km.ci**, **mixlow**, **sdtalt** and **survBayes** were resurrected from the Archive.
- Package **spectrino** was removed from CRAN.
- Package **openNLPmodels** was split into **openNLPmodels.en** and **openNLPmodels.es**.

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Achim Zeileis
WU Wirtschaftsuniversität Wien, Austria
Achim.Zeileis@R-project.org

News from the Bioconductor Project

by the Bioconductor Team

We are pleased to announce Bioconductor 2.5, released on October 28, 2009. Bioconductor 2.5 is compatible with R 2.10.0, and consists of 352 packages. There are 34 new packages, and enhancements to many others. Explore Bioconductor at http://bioconductor.org, and install packages with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite() # install standard packages...
> biocLite("IRanges") # ...or IRanges
```

New and revised packages

This release includes new packages for diverse areas of high-throughput analysis. Highlights include:

Next-generation sequence analysis packages for ChIP-seq (**chipseq**, **ChIPseqR**, **chIPpeakAnno**, **ChIPsim**), differential expression (**DEGseq**, **baySeq**), annotation (**GenomicFeatures**), and image processing (**Rolexa**).

Advanced statistical methods for microarray classification (**BioSeqClass**), differential expression (**cycle**, **LiquidAssociation**, **SpeCond**), and probe reliability (**RPA**).

Microarray domain-specific analysis of copy number, array CGH, tiling (**CNTools**, **CNVtools**, **Starr**, **CGHnormaliter**, **mBPCR**), micro-RNA (**AgiMicroRna**, **RmiR**, **MiChip**), and methylation (**methylumi**) arrays.

Flow cytometry fingerprinting (**flowFP**), cluster merging (**flowMerge**), and plate-based assays (**plateCore**).

Diverse assays related to high-throughput qPCR (**ddCt**, **HTqPCR**), clinical proteomics (**clippda**), and RTCA (**RTCA**).

Integrative tools for data mining (**RTools4TB**), annotation (**GeneAnswers**), network reconstruction (**BUS**), and visualization (**ChromHeatMap**).

Our large collection of microarray- and organism-specific annotation packages have been updated to include information current at the time of the Bioconductor release. These annotation packages contain biological information about microarray probes and the genes they are meant to interrogate, or contain gene-based annotations of whole genomes. They are particularly valuable in providing stable annotations for repeatable research.

Further information on new and existing packages can be found on the Bioconductor web site, which contains 'views' that identify coherent groups of packages. The views link to on-line package descriptions, vignettes, reference manuals, and use statistics.

Other activities

The Bioconductor community met on July 27-28 at our annual conference in Seattle for a combination of scientific talks and hands-on tutorials. The active Bioconductor mailing lists (http://bioconductor.org/docs/mailList.html) connect users with each other, to domain experts, and to maintainers eager to ensure that their packages satisfy the needs of leading edge approaches. Bioconductor package maintainers and the Bioconductor team invest considerable effort in producing high-quality software. The Bioconductor team continues to ensure quality software through technical and scientific reviews of new packages, and daily builds of released packages on Linux, Windows, and Macintosh platforms.

Looking forward

Contributions from the Bioconductor community play an important role in shaping each release. In addition to development of high-quality algorithms for microarray data, we anticipate continued efforts to provide statistically informed analysis of next generation sequence data. Areas of opportunity include the ChIP-seq, RNA-seq, rare variant, and structural variant domains. Analysis of next generation sequence data poses significant challenges in data representation, annotation, and manipulation; the Bioconductor team is actively working on solutions to address these software infrastructure challenges. We also anticipate development of improved graph representations, important for manipulating large networks of biological data. The next release cycle promises to be one of active scientific growth and exploration!

Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center

R Foundation News

by Kurt Hornik

Donations and new members

Donations

Jason Fisher, USA
N. Srinivasan, USA

New benefactors

Tibco Software Inc., Palo Alto, USA
The Generations Network, San Francisco, USA

New supporting members

Stephan Robert Aichele, USA
Matthias Burger, Germany
Jason Fisher, USA
Nicholas Horton, USA
Michael J. Messner, USA
Jacob J. Michaelson, USA
Kenneth S. Spriggs, USA
Vincent Vinh-Hung, Belgium

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org