R News
The Newsletter of the R Project
Volume 4/1, June 2004

Editorial

by *Thomas Lumley*

R has been well accepted in the academic world for some time, but this issue of R News begins with a description of the use of R in a business setting. Marc Schwartz describes how his consulting company, MedAnalytics, came to use R instead of commercial statistical software, and has since provided both code and financial support to the R Project.

The growth of the R community was also evident at the very successful useR! 2004, the first conference for R users. While R aims to blur the distinctions between users and programmers, useR! was definitely different in aims and audience from the previous DSC conferences held in 1999, 2001, and 2003. John Fox provides a summary of the conference in this issue for those who could not make it to Vienna.

A free registration to useR! was the prize in the competition for a new graphic for `http://www.r-project.org`. You will have seen the winning graphic, from Eric Lecoutre, on the way to downloading this newsletter. The graphic is unedited R output (click on it to see the code), showing the power of R for creating complex graphs that are completely reproducible. I would also encourage R users to visit the Journal of Statistical Software (`http:///www.jstatsoft.org`), which publishes peer-reviewed code and documentation. R is heavily represented in recent issues of the journal.

The success of R in penetrating so many statistical niches is due in large part to the wide variety of packages produced by the R community. We have articles describing contributed packages for principal component analysis (`ade4`, used in creating Eric Lecoutre's winning graphic) and statistical process control (`qcc`) and the recommended `survival` package. Jianhua Zhang and Robert Gentlement describe tools from the Bioconductor project that make it easier to explore the resulting variety of code and documentation.

The Programmers' Niche and Help Desk columns both come from guest contributors this time. Gabor Grothendieck,known as frequent poster of answers on the mailing list r-help, has been invited to contribute to the R Help Desk. He presents "Date and Time Classes in R". For the Programmers' Niche, I have written a simple example of classes and methods using the old and new class systems in R. This example arose from a discussion at an R programming course.

Programmers will also find useful Doug Bates' article on least squares calculations. Prompted by discussions on the `r-devel` list, he describes why the simple matrix formulae from so many textbooks are not the best approach either for speed or for accuracy.

Thomas Lumley
Department of Biostatistics
University of Washington, Seattle
`thomas.lumley@R-project.org`

Contents of this issue:

The Decision To Use R

A Consulting Business Perspective

by Marc Schwartz

Preface

The use of R has grown markedly during the past few years, which is a testament to the dedication and commitment of R Core, to the contributions of the growing useR base, and to the variety of disciplines in which R is being used. The recent useR! meeting in Vienna was another strong indicator of growth and was an outstanding success.

In addition to its use in academic areas, R is increasingly being used in non-academic environments, such as government agencies and various businesses, ranging from small firms to large multinational corporations, in a variety of industries.

In a sense, this diversification in the user base is a visible demonstration of the greater comfort that business-based decision makers have regarding the use of Open-Source operating systems, such as Linux, and applications, such as R. Whether these tools and associated support programs are purchased (such as commercial Linux distributions) or are obtained through online collaborative communities, businesses are increasingly recognizing the value of Open-Source software.

In this article, I will describe the use of R (and other Open-Source tools) in a healthcare-based consulting firm, and the value realized by us and, importantly, by our clients through the use of these tools in our company.

Some Background on the Company

MedAnalytics is a healthcare consulting firm located in a suburb of Minneapolis, Minnesota. The company was founded and legally incorporated in August of 2000 to provide a variety of analytic and related services to two principle constituencies. The first are healthcare providers (hospitals and physicians) engaged in quality improvement programs. These clients will typically have existing clinical (as opposed to administrative) databases and wish to use these to better understand the interactions between patient characteristics, clinical processes and outcomes. The second are drug and medical device companies engaged in late-phase and post-market clinical studies. For these clients, MedAnalytics provides services related to protocol and case report form design and development, interim and final analyses and independent data safety monitoring board appointments. In these cases, MedAnalytics will typically partner with healthcare technology and service companies to develop web-based electronic data capture, study management, and reporting. We have worked in many medical specialties including cardiac surgery, cardiology, ophthalmology, orthopaedics, gastroenterology and oncology.

Prior to founding MedAnalytics, I had 5 years of clinical experience in cardiac surgery and cardiology and over 12 years with a medical database software company, where I was principally responsible for the management and analysis of several large multi-site national and regional sub-specialty databases that were sponsored by professional medical societies. My duties included contributing to core dataset design, the generation of periodic aggregate and comparative analyses, the development of multi-variable risk models and contributing to peer-reviewed papers. In that position, the majority of my analyses were performed using SAS (Base, Stat and Graph), though S-PLUS (then from StatSci) was evaluated in the mid-90's and used for a period of time with some smaller datasets.

Evaluating The Tools of the Trade - A Value Based Equation

The founder of a small business must determine how he or she is going to finance initial purchases (of which there are many) and the ongoing operating costs of the company until such time as revenues equal (and, we hope, ultimately exceed) those costs. If financial decisions are not made wisely, a company can find itself rapidly "burning cash" and risking its viability. (It is for this reason that most small businesses fail within the first two or three years.)

Because there are market-driven thresholds of what prospective clients are willing to pay for services, the cost of the tools used to provide client services must be carefully considered. If your costs are too high, you will not recover them via client billings and (despite attitudes that prevailed during the dot.com bubble) you can't make up per-item losses simply by increasing business volume.

An important part of my company's infrastructure would be the applications software that I used. However, it would be foolish for me to base this decision solely on cost. I must consider the "value", which takes into account four key characteristics: cost, time, quality and client service. Like any business, mine would strive to minimize cost and time, while maximizing quality and client service. However, there is a constant tension among these factors and the key would be to find a reasonable balance among them while meeting (and, we hope, exceeding) the needs and expectations of our clients.

First and foremost, a business must present itself and act in a responsible and professional manner and meet legal and regulatory obligations. There would be no value in the software if my clients and I could not trust the results I obtained.

In the case of professional tools, such as software, being used internally, one typically has many alternatives to consider. You must first be sure that the potential choices are validated against a list of required functional and quality benchmarks driven by the needs of your prospective clients. You must, of course, be sure that the choices fit within your operating budget. In the specific case of analytic software applications, the list to consider and the price points were quite varied. At the high end of the price range are applications like SAS and SPSS, which have become prohibitively expensive for small businesses that require commercial licenses. Despite my past lengthy experience with SAS, it would not have been financially practical to consider using it. A single-user desktop license for the three key components (Base, Stat and Graph) would have cost about (US)$5,000 per year, well beyond my budget.

Over a period of several months through late 2000, I evaluated demonstration versions of several statistical applications. At the time I was using Windows XP and thus considered Windows versions of any applications. I narrowed the list to S-PLUS and Stata. Because of my prior experience with both SAS and S-PLUS, I was comfortable with command line applications, as opposed to those with menu-driven "point and click" GUIs. Both S-PLUS and Stata would meet my functional requirements regarding analytic methods and graphics. Because both provided for programming within the language, I could reasonably expect to introduce some productivity enhancements and reduce my costs by more efficient use of my time.

"An Introduction To R"

About then (late 2000), I became aware of R through contacts with other users and through posts to various online forums. I had not yet decided on S-PLUS or Stata so I downloaded and installed R to gain some experience with it. This was circa R version 1.2.x. I began to experiment with R, reading the available online documentation, perusing posts to the r-help list and using some of the books on S-PLUS that I had previously purchased, most notably the first edition (1994) of Venables and Ripley's MASS.

Although I was not yet specifically aware of the R Community and the general culture of the GPL and Open-Source application development, the basic premise and philosophy was not foreign to me. I was quite comfortable with online support mechanisms, having used Usenet, online "knowledge bases" and other online community forums in the past. I had found that, more often that not, these provided more competent and expedient support than did the telephone support from a vendor. I also began to realize that the key factor in the success of my business would be superior client service, and not expensive proprietary technology. My experience and my needs helped me accept the basic notion of Open-Source development and online, community-based support.

From my earliest exposure to R, it was evident to me that this application had evolved substantially in a relatively short period of time (as it has continued to do) under the leadership of some of the best minds in the business. It also became rapidly evident that it would fit my functional requirements and my comfort level with it increased substantially over a period of months. Of course my prior experience with S-PLUS certainly helped me to learn R rapidly, (although there are important differences between S-PLUS and R, as described in the R FAQ).

I therefore postponed any purchase decisions regarding S-PLUS or Stata and made a commitment to using R.

I began to develop and implement various functions that I would need in providing certain analytic and reporting capabilities for clients. Over a period of several weeks, I created a package of functions for internal use that substantially reduced the time it takes to import and structure data from certain clients (typically provided as Excel workbooks or as ASCII files) and then generate a large number of tables and graphs. These types of reports typically use standardized datasets and are created reasonably frequently. Moreover, I could easily enhance and introduce new components to the reports as required by the changing needs of these clients. Over time, I introduced further efficiencies and was able to take a process that initially had taken several days, quite literally down to a few hours. As a result, I was able to spend less time on basic data manipulation and more time on the review, interpretation and description of key findings.

In this way, I could reduce the total time required for a standardized project, reducing my costs and also increasing the value to the client by being able to spend more time on the higher-value services to the client, instead of on internal processes. By enhancing the quality of the product and ultimately passing my cost reductions on to the client, I am able to increase substantially the value of my services.

A Continued Evolution

In my more than three years of using R, I have continued to implement other internal process changes. Most notably, I have moved day-to-day computing operations from Windows XP to Linux. Needless to say, R's support of multiple operating systems made this transition essentially transparent for the application software. I also started using ESS as my primary working environment for R and enjoyed its higher level of support for R compared to the editor that I had used under Windows, which simply provided syntax highlighting.

My transformation to Linux occurred gradually during the later Red Hat (RH) 8.0 beta releases as I became more comfortable with Linux and with the process of replacing Windows functionality that had become "second nature" over my years of MS operating system use (dating back to the original IBM PC/DOS days of 1981). For example, I needed to replace my practice in R of generating WMF files to be incorporated into Word and Powerpoint documents. At first, I switched to generating EPS files for incorporation into OpenOffice.org's (OO.org) Writer and Impress documents, but now I primarily use LaTeX and the "seminar" slide creation package, frequently in conjunction with Sweave. Rather than using Powerpoint or Impress for presentations, I create landscape oriented PDF files and display them with Adobe Acrobat Reader, which supports a full screen presentation mode.

I also use OO.org's Writer for most documents that I need to exchange with clients. For the final version, I export a PDF file from Writer, knowing that clients will have access to Adobe's Reader. Because OO.org's Writer supports MS Word input formats, it provides for interchange with clients when jointly editable documents (or tracking of changes) are required. When clients send me Excel worksheets I use OO.org's Calc which supports Excel data formats (and, in fact, exports much better CSV files from these worksheets than does Excel). I have replaced Outlook with Ximian's Evolution for my e-mail, contact list and calendar management, including the coordination of electronic meeting requests with business partners and clients and, of course, synching my Sony Clie (which Sony recently announced will soon be deprecated in favor of smarter cell phones).

With respect to Linux, I am now using Fedora Core (FC) 2, having moved from RH 8.0 to RH 9 to FC 1. I have in time, replaced all Windows-based applications with Linux-based alternatives, with one exception and that is some accounting software that I need in order to exchange electronic files with my accountant. In time, I suspect that this final hurdle will be removed and with it, the need to use Windows at all. I should make it clear that this is not a goal for philosophic reasons, but for business reasons. I have found that the stability and functionality of the Linux environment has enabled me to engage in a variety of technical and business process related activities that would either be problematic or prohibitively expensive under Windows. This enables me to realize additional productivity gains that further reduce operating costs.

Giving Back

In closing I would like to make a few comments about the R Community.

I have felt fortunate to be a useR and I also have felt strongly that I should give back to the Community - recognizing that ultimately through MedAnalytics and the use of R, I am making a living using software that has been made available through voluntary efforts and which I did not need to purchase.

To that end, over the past three years, I have committed to contributing and giving back to the Community in several ways. First and foremost, by responding to queries on the r-help list and, later, on the r-devel list. There has been, and will continue to be, a continuous stream of new useRs who need basic assistance. As current userRs progress in experience, each of us can contribute to the Community by assisting new useRs. In this way a cycle develops in which the students evolve into teachers.

Secondly, in keeping with the philosophy that "useRs become developers", as I have found the need for particular features and have developed specific solutions, I have made them available via CRAN or the lists. Two functions in particular have been available in the "gregmisc" package since mid-2002, thanks to Greg Warnes. These are the `barplot2()` and `CrossTable()` functions. As time permits, I plan to enhance, most notably, the `CrossTable()` function with a variety of non-parametric correlation measures and will make those available when ready.

Thirdly, in a different area, I have committed to financially contributing back to the Community by supporting the R Foundation and the recent useR! 2004 meeting. This is distinct from volunteering my time and recognizes that, ultimately, even a voluntary Community such as this one has some operational costs to cover. When, in 2000-2001, I was contemplating the purchase of S-PLUS and Stata, these would have cost approximately (US)$2,500 and (US)$1,200 respectively. In effect, I have committed those dollars to supporting the R Foundation at the Benefactor level over the coming years.

I would challenge and urge other commercial useRs to contribute to the R Foundation at whatever level makes sense for your organization.

Thanks

It has been my pleasure and privilege to be a member of this Community over the past three years. It has been nothing short of phenomenal to experience the growth of this Community during that time.

I would like to thank Doug Bates for his kind invitation to contribute this article, the idea for which arose in a discussion we had during the recent useR! meeting in Vienna. I would also like to extend my thanks to R Core and to the R Community at large for their continued drive to create and support this wonderful vehicle for international collaboration.

Marc Schwartz
MedAnalytics, Inc., Minneapolis, Minnesota, USA
MSchwartz@MedAnalytics.com

The ade4 package - I : One-table methods

by Daniel Chessel, Anne B Dufour and Jean Thioulouse

Introduction

This paper is a short summary of the main classes defined in the ade4 package for one table analysis methods (e.g., principal component analysis). Other papers will detail the classes defined in ade4 for two-tables coupling methods (such as canonical correspondence analysis, redundancy analysis, and co-inertia analysis), for methods dealing with K-tables analysis (i.e., three-ways tables), and for graphical methods.

This package is a complete rewrite of the ADE4 software (Thioulouse et al. (1997), http://pbil.univ-lyon1.fr/ADE-4/) for the R environment. It contains **D**ata **A**nalysis functions to analyse **E**cological and **E**nvironmental data in the framework of **E**uclidean **E**xploratory methods, hence the name **ade4** (i.e., 4 is not a version number but means that there are four E in the acronym).

The ade4 package is available in CRAN, but it can also be used directly online, thanks to the Rweb system (`http://pbil.univ-lyon1.fr/Rweb/`). This possibility is being used to provide multivariate analysis services in the field of bioinformatics, particularly for sequence and genome structure analysis at the PBIL (`http://pbil.univ-lyon1.fr/`). An example of these services is the automated analysis of the codon usage of a set of DNA sequences by correspondence analysis (`http://pbil.univ-lyon1.fr/mva/coa.php`).

The duality diagram class

The basic tool in ade4 is the duality diagram Escoufier (1987). A duality diagram is simply a list that contains a triplet ($\mathbf{X}$, $\mathbf{Q}$, $\mathbf{D}$):

- $\mathbf{X}$ is a table with n rows and p columns, considered as p points in $\mathbb{R}^n$ (column vectors) or n points in $\mathbb{R}^p$ (row vectors).

- $\mathbf{Q}$ is a $p \times p$ diagonal matrix containing the weights of the p columns of $\mathbf{X}$, and used as a scalar product in $\mathbb{R}^p$ ($\mathbf{Q}$ is stored under the form of a vector of length p).

- $\mathbf{D}$ is a $n \times n$ diagonal matrix containing the weights of the n rows of $\mathbf{X}$, and used as a scalar product in $\mathbb{R}^n$ ($\mathbf{D}$ is stored under the form of a vector of length n).

For example, if $\mathbf{X}$ is a table containing normalized quantitative variables, if $\mathbf{Q}$ is the identity matrix $\mathbf{I}_p$ and if $\mathbf{D}$ is equal to $\frac{1}{n}\mathbf{I}_n$, the triplet corresponds to a principal component analysis on correlation matrix (normed PCA). Each basic method corresponds to a particular triplet (see table 1), but more complex methods can also be represented by their duality diagram.

Functions	Analyses	Notes
dudi.pca	principal component	1
dudi.coa	correspondence	2
dudi.acm	multiple correspondence	3
dudi.fca	fuzzy correspondence	4
dudi.mix	analysis of a mixture of numeric and factors	5
dudi.nsc	non symmetric correspondence	6
dudi.dec	decentered correspondence	7

The dudi functions. 1: Principal component analysis, same as prcomp/princomp. 2: Correspondence analysis Greenacre (1984). 3: Multiple correspondence analysis Tenenhaus and Young (1985). 4: Fuzzy correspondence analysis Chevenet et al. (1994). 5: Analysis of a mixture of numeric variables and factors Hill and Smith (1976), Kiers (1994). 6: Non symmetric correspondence analysis Kroonenberg and Lombardo (1999). 7: Decentered correspondence analysis Dolédec et al. (1995).

The singular value decomposition of a triplet gives principal axes, principal components, and row and column coordinates, which are added to the triplet for later use.

We can use for example a well-known dataset from the base package :

```
> data(USArrests)
> pca1 <- dudi.pca(USArrests,scannf=FALSE,nf=3)
```

`scannf = FALSE` means that the number of principal components that will be used to compute row and column coordinates should not be asked interactively to the user, but taken as the value of argument `nf` (by default, `nf = 2`). Other parameters allow to choose between centered, normed or raw PCA (default is centered and normed), and to set arbitrary row and column weights. The `pca1` object is a duality diagram, i.e., a list made of several vectors and dataframes:

```
> pca1
Duality diagramm
class: pca dudi
$call: dudi.pca(df = USArrests, scannf = FALSE, nf=3)

$nf: 3 axis-components saved
$rank: 4
eigen values: 2.48 0.9898 0.3566 0.1734
  vector length mode    content
1 $cw    4      numeric column weights
2 $lw    50     numeric row weights
3 $eig   4      numeric eigen values

  data.frame nrow ncol content
1 $tab       50   4    modified array
2 $li        50   3    row coordinates
3 $l1        50   3    row normed scores
4 $co        4    3    column coordinates
5 $c1        4    3    column normed scores
other elements: cent norm
```

pca1$lw and pca1$cw are the row and column weights that define the duality diagram, together with the data table (pca1$tab). pca1$eig contains the eigenvalues. The row and column coordinates are stored in pca1$li and pca1$co. The variance of these coordinates is equal to the corresponding eigenvalue, and unit variance coordinates are stored in pca1$l1 and pca1$c1 (this is usefull to draw biplots).

The general optimization theorems of data analysis take particular meanings for each type of analysis, and graphical functions are proposed to draw the *canonical graphs*, i.e., the graphical expression corresponding to the mathematical property of the object. For example, the normed PCA of a quantitative variable table gives a score that maximizes the sum of squared correlations with variables. The PCA canonical graph is therefore a graph showing how the sum of squared correlations is maximized for the variables of the data set. On the USArrests example, we obtain the following graphs:

```
> score(pca1)
> s.corcircle(pca1$co)
```

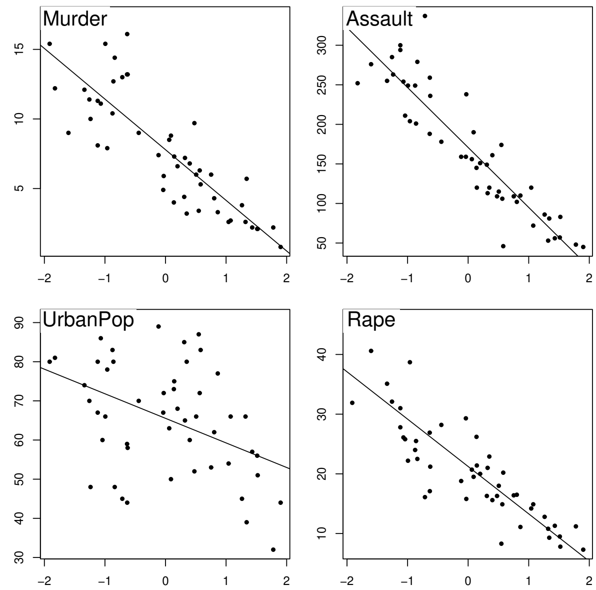

Figure 1: *One dimensional canonical graph for a normed PCA. Variables are displayed as a function of row scores, to get a picture of the maximization of the sum of squared correlations.*

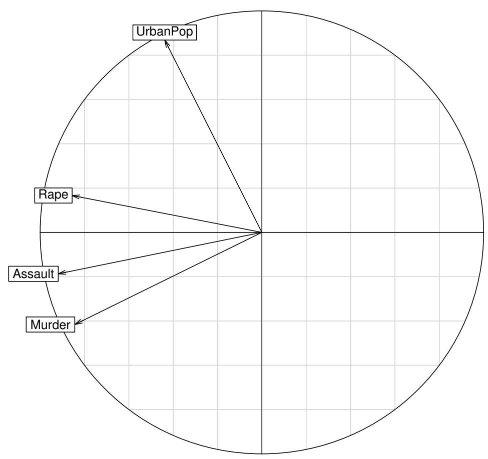

Figure 2: *Two dimensional canonical graph for a normed PCA (correlation circle): the direction and length of arrows show the quality of the correlation between variables and between variables and principal components.*

The scatter function draws the biplot of the PCA (i.e., a graph with both rows and columns superimposed):

```
> scatter(pca1)
```

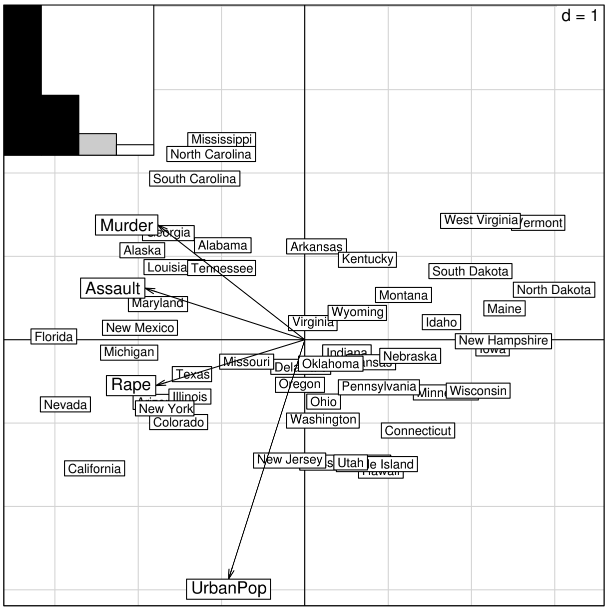

Figure 3: *The PCA biplot. Variables are symbolized by arrows and they are superimposed to the individuals display. The scale of the graph is given by a grid, which size is given in the upper right corner. Here, the length of the side of grid squares is equal to one. The eigenvalues bar chart is drawn in the upper left corner, with the two black bars corresponding to the two axes used to draw the biplot. Grey bars correspond to axes that were kept in the analysis, but not used to draw the graph.*

Separate factor maps can be drawn with the `s.corcircle` (see figure 2) and `s.label` functions:

```
> s.label(pca1$li)
```

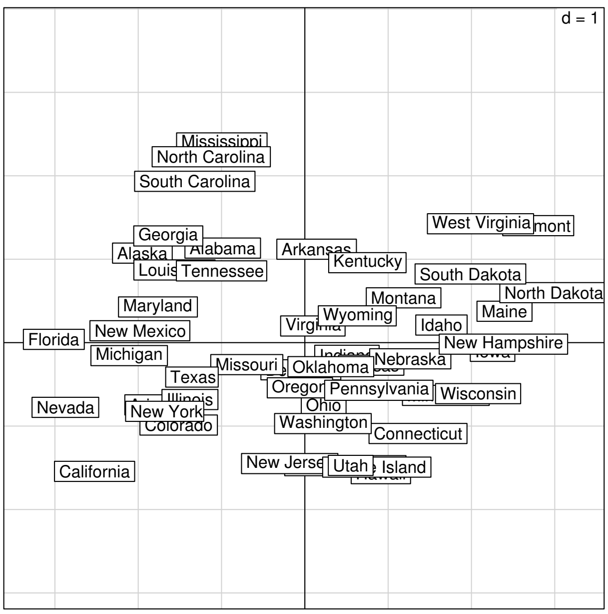

Figure 4: *Individuals factor map in a normed PCA.*

Distance matrices

A duality diagram can also come from a distance matrix, if this matrix is Euclidean (i.e., if the distances in the matrix are the distances between some points in a Euclidean space). The ade4 package contains functions to compute dissimilarity matrices (`dist.binary` for binary data, and `dist.prop` for frequency data), test whether they are Euclidean Gower and Legendre (1986), and make them Euclidean (`quasieuclid`, `lingoes`, Lingoes (1971), `cailliez`, Cailliez (1983)). These functions are useful to ecologists who use the works of Legendre and Anderson (1999) and Legendre and Legendre (1998).

The Yanomama data set (Manly (1991)) contains three distance matrices between 19 villages of Yanomama Indians. The `dudi.pco` function can be used to compute a principal coordinates analysis (PCO, Gower (1966)), that gives a Euclidean representation of the 19 villages. This Euclidean representation allows to compare the geographical, genetic and anthropometric distances.

```
> data(yanomama)
> gen <- quasieuclid(as.dist(yanomama$gen))
> geo <- quasieuclid(as.dist(yanomama$geo))
> ant <- quasieuclid(as.dist(yanomama$ant))
> geo1 <- dudi.pco(geo, scann = FALSE, nf = 3)
> gen1 <- dudi.pco(gen, scann = FALSE, nf = 3)
> ant1 <- dudi.pco(ant, scann = FALSE, nf = 3)
> par(mfrow=c(2,2))
> scatter(geo1)
> scatter(gen1)
> scatter(ant1,posi="bottom")
```

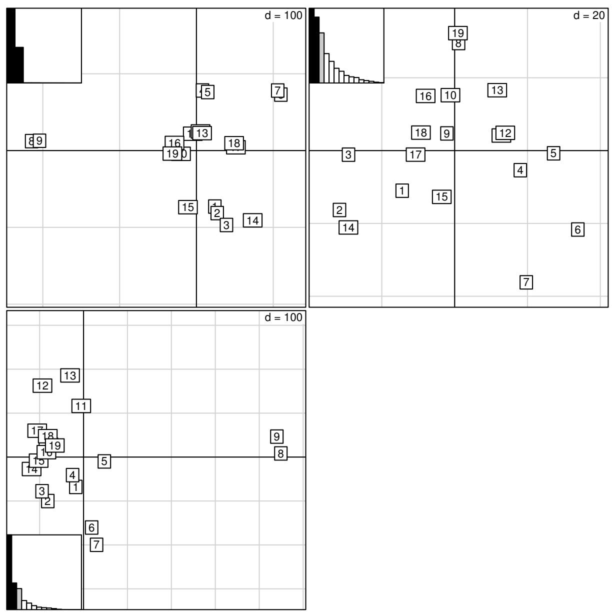

Figure 5: *Comparison of the PCO analysis of the three distance matrices of the Yanomama data set: geographic, genetic and anthropometric distances.*

Taking into account groups of individuals

In sites x species tables, rows correspond to sites, columns correspond to species, and the values are the number of individuals of species j found at site i. These tables can have many columns and cannot be used in a discriminant analysis. In this case, between-class analyses (`between` function) are a better alternative, and they can be used with any duality diagram. The between-class analysis of triplet ($\mathbf{X}$, $\mathbf{Q}$, $\mathbf{D}$) for a given factor $\mathbf{f}$ is the analysis of the triplet ($\mathbf{G}$, $\mathbf{Q}$, $\mathbf{D}_w$), where $\mathbf{G}$ is the table of the means of table $\mathbf{X}$ for the groups defined by $\mathbf{f}$, and $\mathbf{D}_w$ is the diagonal matrix of group weights. For example, a between-class correspondence analysis (BCA) is very simply obtained after a correspondence analysis (CA):

```
> data(meaudret)
> coa1<-dudi.coa(meaudret$fau, scannf = FALSE)
> bet1<-between(coa1,meaudret$plan$sta,
+ scannf=FALSE)
> plot(bet1)
```

The `meaudret$fau` dataframe is an ecological table with 24 rows corresponding to six sampling sites along a small French stream (the Meaudret). These six sampling sites were sampled four times (spring, summer, winter and autumn), hence the 24 rows. The 13 columns correspond to 13 ephemerotera species. The CA of this data table is done with the `dudi.coa` function, giving the `coa1` duality diagram. The corresponding bewteen-class analysis is done with the `between` function, considering the sites as classes (`meaudret$plan$sta` is a factor defining the classes). Therefore, this is a between-sites analysis, which aim is to discriminate the sites, given the distribution of ephemeroptera species. This gives the `bet1` duality diagram, and Figure 6 shows the graph obtained by plotting this object.

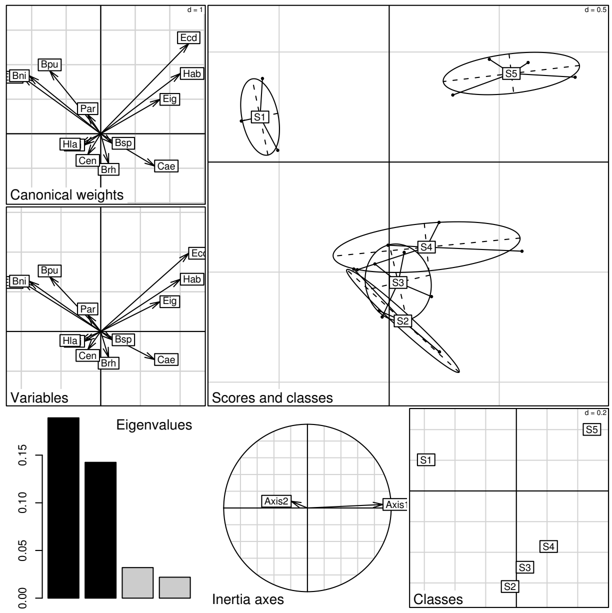

Figure 6: *BCA plot. This is a composed plot, made of : 1- the species canonical weights (top left), 2- the species scores (middle left), 3- the eigenvalues bar chart (bottom left), 4- the plot of plain CA axes projected into BCA (bottom center), 5- the gravity centers of classes (bottom right), 6- the projection of the rows with ellipses and gravity center of classes (main graph).*

Like between-class analyses, linear discriminant analysis (`discrimin` function) can be extended to any duality diagram ($\mathbf{G}$, $(\mathbf{X}^t\mathbf{D}\mathbf{X})^-$, $\mathbf{D}_w$), where $(\mathbf{X}^t\mathbf{D}\mathbf{X})^-$ is a generalized inverse. This gives for example a correspondence discriminant analysis (Perrière et al. (1996), Perrière and Thioulouse (2002)), that can be computed with the `discrimin.coa` function.

Opposite to between-class analyses are within-class analyses, corresponding to diagrams ($\mathbf{X} - \mathbf{Y}\mathbf{D}_w^{-1}\mathbf{Y}^t\mathbf{D}\mathbf{X}$, $\mathbf{Q}$, $\mathbf{D}$) (`within` functions). These analyses extend to any type of variables the Multiple Group Principal Component Analysis (MGPCA, Thorpe (1983a), Thorpe (1983b), Thorpe and Leamy (1983). Furthermore, the `within.coa` function introduces the double within-class correspondence analysis (also named internal correspondence analysis, Cazes et al. (1988)).

Permutation tests

Permutation tests (also called Monte-Carlo tests, or randomization tests) can be used to assess the statistical significance of between-class analyses. Many permutation tests are available in the ade4 package, for example `mantel.randtest`, `procuste.randtest`, `randtest.between`, `randtest.coinertia`, `RV.rtest`, `randtest.discrimin`, and several of these tests are available both in R (`mantel.rtest`) and in C (`mantel.randtest`) programming langage. The R version allows to see how computations are performed, and to write easily other tests, while the C version is needed for performance reasons.

The statistical significance of the BCA can be evaluated with the randtest.between function. By default, 999 permutations are simulated, and the resulting object (test1) can be plotted (Figure 7). The p-value is highly significant, which confirms the existence of differences between sampling sites. The plot shows that the observed value is very far to the right of the histogram of simulated values.

```
> test1<-randtest.between(bet1)
> test1
Monte-Carlo test
Observation: 0.4292
Call: randtest.between(xtest = bet1)
Based on 999 replicates
Simulated p-value: 0.001
> plot(test1,main="Between class inertia")
```

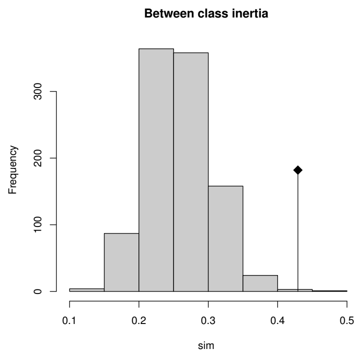

Figure 7: *Histogram of the 999 simulated values of the randomization test of the* bet1 *BCA. The observed value is given by the vertical line, at the right of the histogram.*

Conclusion

We have described only the most basic functions of the ade4 package, considering only the simplest one-table data analysis methods. Many other dudi methods are available in ade4, for example multiple correspondence analysis (dudi.acm), fuzzy correspondence analysis (dudi.fca), analysis of a mixture of numeric variables and factors (dudi.mix), non symmetric correspondence analysis (dudi.nsc), decentered correspondence analysis (dudi.dec).

We are preparing a second paper, dealing with two-tables coupling methods, among which canonical correspondence analysis and redundancy analysis are the most frequently used in ecology (Legendre and Legendre (1998)). The ade4 package proposes an alternative to these methods, based on the co-inertia criterion (Dray et al. (2003)).

The third category of data analysis methods available in ade4 are K-tables analysis methods, that try to extract the stable part in a series of tables. These methods come from the STATIS strategy, Lavit et al. (1994) (statis and pta functions) or from the multiple coinertia strategy (mcoa function). The mfa and foucart functions perform two variants of K-tables analysis, and the STATICO method (function ktab.match2ktabs, Thioulouse et al. (2004)) allows to extract the stable part of species-environment relationships, in time or in space.

Methods taking into account spatial constraints (multispati function) and phylogenetic constraints (phylog function) are under development.

Bibliography

Cailliez, F. (1983). The analytical solution of the additive constant problem. *Psychometrika*, 48:305–310. 7

Cazes, P., Chessel, D., and Dolédec, S. (1988). L'analyse des correspondances internes d'un tableau partitionné : son usage en hydrobiologie. *Revue de Statistique Appliquée*, 36:39–54. 8

Chevenet, F., Dolédec, S., and Chessel, D. (1994). A fuzzy coding approach for the analysis of long-term ecological data. *Freshwater Biology*, 31:295–309. 5

Dolédec, S., Chessel, D., and Olivier, J. (1995). L'analyse des correspondances décentrée: application aux peuplements ichtyologiques du haut-rhône. *Bulletin Français de la Pêche et de la Pisciculture*, 336:29–40. 5

Dray, S., Chessel, D., and Thioulouse, J. (2003). Co-inertia analysis and the linking of ecological tables. *Ecology*, 84:3078–3089. 9

Escoufier, Y. (1987). The duality diagramm : a means of better practical applications. In Legendre, P. and Legendre, L., editors, *Development in numerical ecology*, pages 139–156. NATO advanced Institute , Serie G .Springer Verlag, Berlin. 5

Gower, J. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338. 7

Gower, J. and Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48. 7

Greenacre, M. (1984). *Theory and applications of correspondence analysis*. Academic Press, London. 5

Hill, M. and Smith, A. (1976). Principal component analysis of taxonomic data with multi-state discrete characters. *Taxon*, 25:249–255. 5

Kiers, H. (1994). Simple structure in component analysis techniques for mixtures of qualitative ans quantitative variables. *Psychometrika*, 56:197–212. 5

Kroonenberg, P. and Lombardo, R. (1999). Non-symmetric correspondence analysis: a tool for analysing contingency tables with a dependence structure. *Multivariate Behavioral Research*, 34:367–396. 5

Lavit, C., Escoufier, Y., Sabatier, R., and Traissac, P. (1994). The act (statis method). *Computational Statistics and Data Analysis*, 18:97–119. 9

Legendre, P. and Anderson, M. (1999). Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. *Ecological Monographs*, 69:1–24. 7

Legendre, P. and Legendre, L. (1998). *Numerical ecology*. Elsevier Science BV, Amsterdam, 2nd english edition edition. 7, 9

Lingoes, J. (1971). Somme boundary conditions for a monotone analysis of symmetric matrices. *Psychometrika*, 36:195–203. 7

Manly, B. F. J. (1991). *Randomization and Monte Carlo methods in biology*. Chapman and Hall, London. 7

Perrière, G., Combet, C., Penel, S., Blanchet, C., Thioulouse, J., Geourjon, C., Grassot, J., Charavay, C., Gouy, M. Duret, L., and Deléage, G. (2003). Integrated databanks access and sequence/structure analysis services at the pbil. *Nucleic Acids Research*, 31:3393–3399.

Perrière, G., Lobry, J., and Thioulouse, J. (1996). Correspondence discriminant analysis: a multivariate method for comparing classes of protein and nucleic acid sequences. *Computer Applications in the Biosciences*, 12:519–524. 8

Perrière, G. and Thioulouse, J. (2002). Use of correspondence discriminant analysis to predict the subcellular location of bacterial proteins. *Computer Methods and Programs in Biomedicine*, in press. 8

Tenenhaus, M. and Young, F. (1985). An analysis and synthesis of multiple correspondence analysis, optimal scaling, dual scaling, homogeneity analysis ans other methods for quantifying categorical multivariate data. *Psychometrika*, 50:91–119. 5

Thioulouse, J., Chessel, D., Dolédec, S., and Olivier, J. (1997). Ade-4: a multivariate analysis and graphical display software. *Statistics and Computing*, 7:75–83. 5

Thioulouse, J., Simier, M., and Chessel, D. (2004). Simultaneous analysis of a sequence of paired ecological tables. *Ecology*, 85:272–283. 9

Thorpe, R. S. (1983a). A biometric study of the effects of growth on the analysis of geographic variation: Tooth number in green geckos (reptilia: Phelsuma). *Journal of Zoology*, 201:13–26. 8

Thorpe, R. S. (1983b). A review of the numerical methods for recognising and analysing racial differentiation. In Felsenstein, J., editor, *Numerical Taxonomy*, Series G, pages 404–423. Springer-Verlag, New York. 8

Thorpe, R. S. and Leamy, L. (1983). Morphometric studies in inbred house mice (mus sp.): Multivariate analysis of size and shape. *Journal of Zoology*, 199:421–432. 8

qcc: An R package for quality control charting and statistical process control

by Luca Scrucca

Introduction

The **qcc** package for the R statistical environment allows to:

- plot Shewhart quality control charts for continuous, attribute and count data;
- plot Cusum and EWMA charts for continuous data;
- draw operating characteristic curves;
- perform process capability analyses;
- draw Pareto charts and cause-and-effect diagrams.

I started writing the package to provide the students in the class I teach a tool for learning the basic concepts of statistical quality control, at the introductory level of a book such as Montgomery (2000). Due to the intended final users, a free statistical environment was important and R was a natural candidate.

The initial design, albeit it was written from scratch, reflected the set of functions available in S-Plus. Users of this last environment will find some similarities but also differences as well. In particular, during the development I added more tools and re-designed most of the library to meet the requirements I thought were important to satisfy.

Creating a `qcc` object

A `qcc` object is the basic building block to start with. This can be simply created invoking the `qcc` function, which in its simplest form requires two arguments: a data frame, a matrix or a vector containing the observed data, and a string value specifying the control chart to be computed. Standard Shewhart control charts are available. These are often classified according to the type of quality characteristic that they are supposed to monitor (see Table 1).

Control charts for continuous variables are usually based on several samples with observations collected at different time point. Each sample or "rationale group" must be provided as a row of a data frame or a matrix. The function `qcc.groups` can be used to easily group a vector of data values based on a sample indicator. Unequal sample sizes are allowed.

Suppose we have extracted samples for a characteristic of interest from an ongoing production process:

```
> data(pistonrings)
> attach(pistonrings)
> dim(pistonrings)
[1] 200   3
> pistonrings
    diameter sample trial
1     74.030      1  TRUE
2     74.002      1  TRUE
3     74.019      1  TRUE
4     73.992      1  TRUE
5     74.008      1  TRUE
6     73.995      2  TRUE
7     73.992      2  TRUE
...
199   74.000     40 FALSE
200   74.020     40 FALSE
> diameter <- qcc.groups(diameter, sample)
> dim(diameter)
[1] 40  5
> diameter
     [,1]   [,2]   [,3]   [,4]   [,5]
1  74.030 74.002 74.019 73.992 74.008
2  73.995 73.992 74.001 74.011 74.004
...
40 74.010 74.005 74.029 74.000 74.020
```

Using the first 25 samples as training data, an $\overline{X}$ chart can be obtained as follows:

```
> obj <- qcc(diameter[1:25,], type="xbar")
```

By default a Shewhart chart is drawn (see Figure 1) and an object of class 'qcc' is returned. Summary statistics can be retrieved using the `summary` method (or simply by printing the object):

```
> summary(obj)

Call:
qcc(data = diameter[1:25, ], type = "xbar")

xbar chart for diameter[1:25, ]

Summary of group statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  73.99   74.00   74.00   74.00   74.00   74.01

Group sample size:  5
Number of groups:  25
Center of group statistics:  74.00118
Standard deviation:  0.009887547

Control limits:
      LCL      UCL
 73.98791 74.01444
```

type	Control charts for variables	
"xbar"	$\overline{X}$ chart	Sample means are plotted to control the mean level of a continuous process variable.
"xbar.one"	$\overline{X}$ chart	Sample values from a one–at–time data process to control the mean level of a continuous process variable.
"R"	R chart	Sample ranges are plotted to control the variability of a continuous process variable.
"S"	S chart	Sample standard deviations are plotted to control the variability of a continuous process variable.
type	Control charts for attributes	
"p"	p chart	The proportion of nonconforming units is plotted. Control limits are based on the binomial distribution.
"np"	np chart	The number of nonconforming units is plotted. Control limits are based on the binomial distribution.
"c"	c chart	The number of defectives per unit are plotted. This chart assumes that defects of the quality attribute are rare, and the control limits are computed based on the Poisson distribution.
"u"	u chart	The average number of defectives per unit is plotted. The Poisson distribution is used to compute control limits, but, unlike the c chart, this chart does not require a constant number of units.

Table 1: Shewhart control charts available in the **qcc** package.

The number of groups and their sizes are reported in this case, whereas a table is provided in the case of unequal sample sizes. Moreover, the center of group statistics (the overall mean for an $\overline{X}$ chart) and the within-group standard deviation of the process are returned.

The main goal of a control chart is to monitor a process. If special causes of variation are present the process is said to be "out of control" and actions are to be taken to find, and possibly eliminate, such causes. A process is declared to be "in control" if all points charted lie randomly within the control limits. These are usually computed at $\pm 3\sigma$s from the center. This default can be changed using the argument `nsigmas` (so called "american practice") or specifying the confidence level (so called "british practice") through the `confidence.level` argument.

Assuming a Gaussian distribution is appropriate for the statistic charted, the two practices are equivalent, since limits at $\pm 3\sigma$s correspond to a two-tails probability of 0.0027 under a standard normal curve.

Instead of using standard calculations to compute the group statistics and the corresponding control limits (see Montgomery (2000), Wetherill and Brown (1991)), the center, the within-group standard deviation and the control limits may be specified directly by the user through the arguments `center`, `std.dev` and `limits`, respectively.

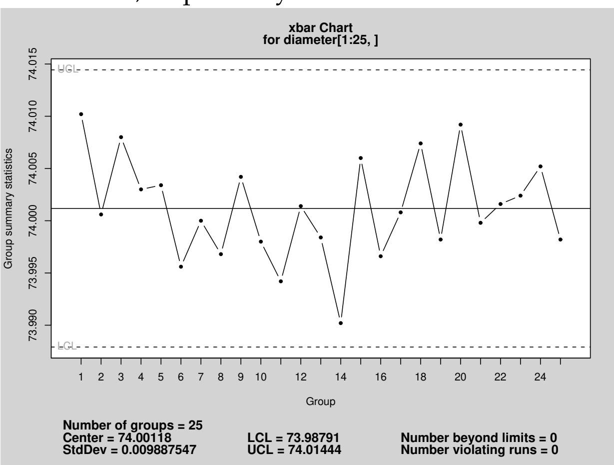

Figure 1: A $\overline{X}$ chart for samples data.

Shewhart control charts

A Shewhart chart is automatically plotted when an object of class 'qcc' is created, unless the qcc function is called with the argument `plot=FALSE`. The method responsible for plotting a control chart can however be invoked directly as follows:

```
> plot(obj)
```

giving the plot in Figure 1. This control chart has the center line (horizontal solid line), the upper and lower control limits (dashed lines), and the sample group statistics are drawn as points connected with lines. Unless we provide the argument `add.stats=FALSE`, at the bottom of the plot some summary statistics are shown, together with the number of points beyond control limits and the number of violating runs (a run has length 5 by default).

Once a process is considered to be "in-control", we may use the computed limits for monitoring new data sampled from the same ongoing process. For example,

```
> obj <- qcc(diameter[1:25,], type="xbar",
+             newdata=diameter[26:40,])
```

plot the $\overline{X}$ chart for both calibration data and new data (see Figure 2), but all the statistics and the control limits are solely based on the first 25 samples.

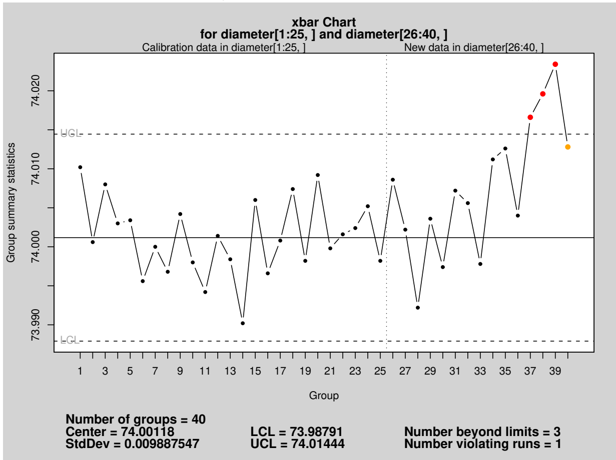

Figure 2: A $\overline{X}$ chart with calibration data and new data samples.

If only the new data need to be plotted we may provide the argument `chart.all=FALSE` in the call to the qcc function, or through the plotting method:

```
> plot(obj, chart.all=FALSE)
```

Many other arguments may be provided in the call to the qcc function and to the corresponding plotting method, and we refer the reader to the on–line documentation for details (`help(qcc)`).

As reported on Table 1, an $\overline{X}$ chart for one–at–time data may be obtained specifying `type="xbar.one"`. In this case the data charted are simply the sample values, and the within-group standard deviation is estimated by moving ranges of k (by default 2) points.

Control charts for attributes mainly differ from the previous examples in that we need to provide sample sizes through the `size` argument (except for the c chart). For example, a p chart can be obtained as follows:

```
> data(orangejuice)
> attach(orangejuice)
```

```
> orangejuice
   sample  D size trial
1       1 12   50  TRUE
2       2 15   50  TRUE
3       3  8   50  TRUE
...
53     53  3   50 FALSE
54     54  5   50 FALSE
> obj2 <- qcc(D[trial], sizes=size[trial], type="p")
```

where we use the logical indicator variable `trial` to select the first 30 samples as calibration data.

Operating characteristic function

An operating characteristic (OC) curve provides information about the probability of not detecting a shift in the process. This is usually referred to as the type II error, that is, the probability of erroneously accepting a process as being "in control".

The OC curves can be easily obtained from an object of class ‘`qcc`’:

```
> par(mfrow=c(1,2))
> oc.curves(obj)
> oc.curves(obj2)
```

The function `oc.curves` invisibly returns a matrix or a vector of probabilities values for the type II error. More arguments are available (see `help(oc.curves)`), in particular `identify=TRUE` allows to interactively identify values on th e plot.

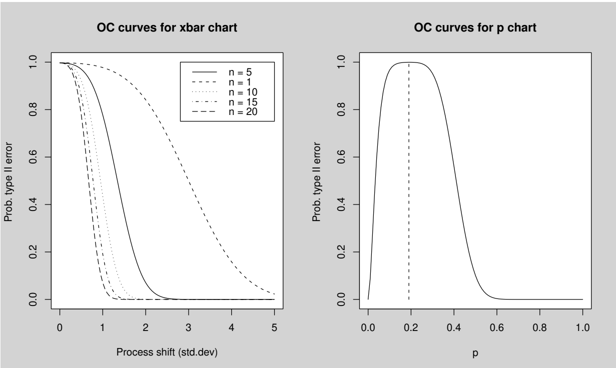

Figure 3: Operating characteristics curves.

Cusum charts

Cusum charts display how the group summary statistics deviate above or below the process center or target value, relative to the standard error of the summary statistics. They are useful to detect small and permanent variation on the mean of the process.

The basic cusum chart implemented in the **qcc** package is only available for continuous variables at the moment. Assuming a ‘`qcc`’ object has been created, we can simply input the following code:

```
> cusum(obj)
```

and the resulting cusum chart is shown in Figure 4. This displays on a single plot both the positive deviations and the negative deviations from the target, which by default is set to the center of group statistics. If a different target for the process is required, this value may be provided through the argument `target` when creating the object of class ‘`qcc`’. Two further aspects may need to be controlled. The argument `decision.int` can be used to specifies the number of standard errors of the summary statistics at which the cumulative sum displays an out of control; by default it is set at 5. The amount of shift in the process to be detected by the cusum chart, measured in standard errors of the summary statistics, is controlled by the argument `se.shift`, by default set at 1.

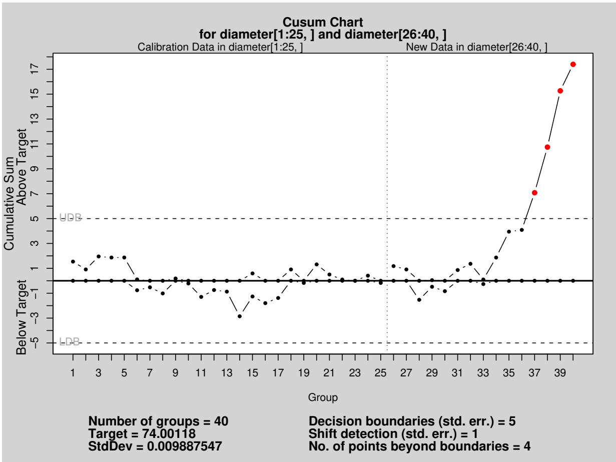

Figure 4: Cusum chart.

EWMA charts

Exponentially weighted moving average (EWMA) charts smooth a series of data based on a moving average with weights which decay exponentially. As for the cusum chart, also this plot can be used to detect small and permanent variation on the mean of the process.

The EWMA chart depends on the smoothing parameter λ, which controls the weighting scheme applied. By default it is set at 0.2, but it can be modified using the argument `lambda`. Assuming again that a ‘`qcc`’ object has been created, an EWMA chart can be obtained as follows:

```
> ewma(obj)
```

and the resulting graph is shown in Figure 5.

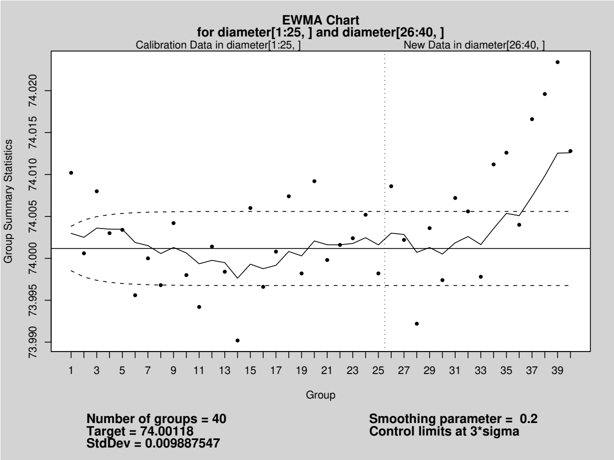

Figure 5: EWMA chart.

Process capability analysis

Process capability indices for a characteristic of interest from a continuous process can be obtained through the function `process.capability`. This takes as mandatory arguments an object of class ‘`qcc`’ for a `"xbar"` or `"xbar.one"` type, and `spec.limits`, a vector specifying the lower (LSL) and upper (USL) specification limits. For example, using the previously created ‘`qcc`’ object, we may simply use:

```
> process.capability(obj,
+                    spec.limits=c(73.95,74.05))

Process Capability Analysis

Call:
process.capability(object = obj,
                   spec.limits = c(73.95, 74.05))

Number of obs = 125          Target = 74
       Center = 74.00118        LSL = 73.95
       StdDev = 0.009887547     USL = 74.05

Capability indices:

      Value   2.5%  97.5%
Cp    1.686  1.476  1.895
Cp_l  1.725  1.539  1.912
Cp_u  1.646  1.467  1.825
Cp_k  1.646  1.433  1.859
Cpm   1.674  1.465  1.882

Exp<LSL 0%   Obs<LSL 0%
Exp>USL 0%   Obs>USL 0%
```

The graph shown in Figure 6 is an histogram of data values, with vertical dotted lines for the upper and the lower specification limits. The target is shown as a vertical dashed line and its value can be provided using the `target` argument in the call; if missing, the value from the ‘`qcc`’ object is used if available, otherwise the target is set at the middle value between the specification limits.

The dashed curve is a normal density for a Gaussian distribution matching the observed mean and standard deviation. The latter is estimated using the within-group standard deviation, unless a value is specified in the call using the `std.dev` argument.

The capability indices reported on the graph are also printed at console, together with confidence intervals computed at the level specified by the `confidence.level` argument (by default set at 0.95).

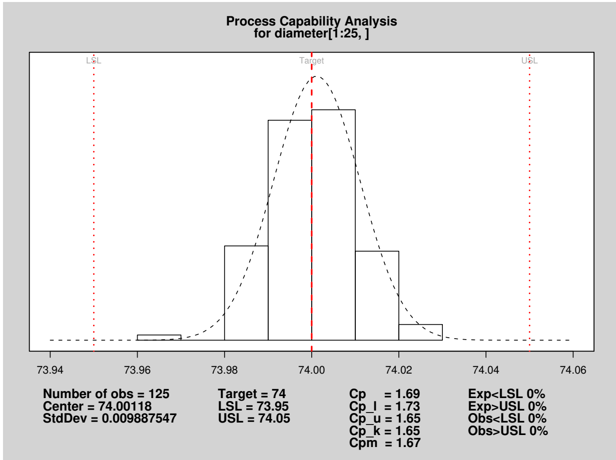

Figure 6: Process capability analysis

A further display, called “process capability sixpack plots”, is also available. This is a graphical summary formed by plotting on the same graphical device:

- a $\overline{X}$ chart
- a R or S chart (if sample sizes > 10)
- a run chart
- a histogram with specification limits
- a normal Q–Q plot
- a capability plot

As an example we may input the following code:

```
> process.capability.sixpack(obj,
+                  spec.limits=c(73.95,74.05),
+                  target= 74.01)
```

Pareto charts and cause-and-effect diagrams

A Pareto chart is a barplot where the categories are ordered using frequencies in non increasing order, with a line added to show their cumulative sum. Pareto charts are useful to identify those factors that have the greatest cumulative effect on the system, and thus screen out the less significant factors in an analysis.

A simple Pareto chart may be drawn using the function `pareto.chart`, which in its simpler form requires a vector of values. For example:

```
> defect <- c(80, 27, 66, 94, 33)
> names(defect) <-
+   c("price code", "schedule date",
+     "supplier code", "contact num.",
+     "part num.")
> pareto.chart(defect)
```

The function returns a table of descriptive statistics and the Pareto chart shown in Figure 7. More arguments can be provided to control axes labels and limits, the main title and bar colors (see `help(pareto.chart)`).

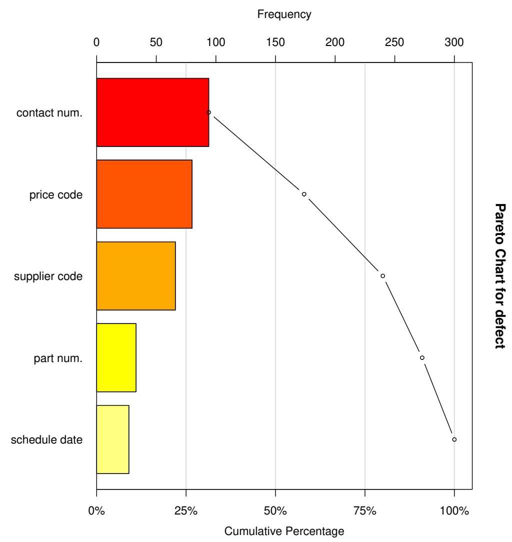

Figure 7: Pareto chart.

A cause-and-effect diagram, also called an Ishikawa diagram (or fish bone diagram), is used to associate multiple possible causes with a single effect. Thus, given a particular effect, the diagram is constructed to identify and organize possible causes for it.

A very basic implementation of this type of diagram is available in the **qcc** package. The function `cause.and.effect` requires at least two arguments: `cause`, a list of causes and branches providing descriptive labels, and `effect`, a string describing the effect. Other arguments are available, such as `cex` and `font` which control, respectively, the character expansions and the fonts used for labeling branches, causes and the effect. The following example creates the diagram shown in Figure 8.

```
> cause.and.effect(
+   cause=list(Measurements=c("Microscopes",
+                             "Inspectors")
+              Materials=c("Alloys",
+                          "Suppliers"),
+              Personnel=c("Supervisors",
+                          "Operators"),
+              Environment=c("Condensation",
+                            "Moisture"),
+              Methods=c("Brake", "Engager",
+                        "Angle"),
+              Machines=c("Speed", "Bits",
+                         "Sockets")),
+   effect="Surface Flaws")
```

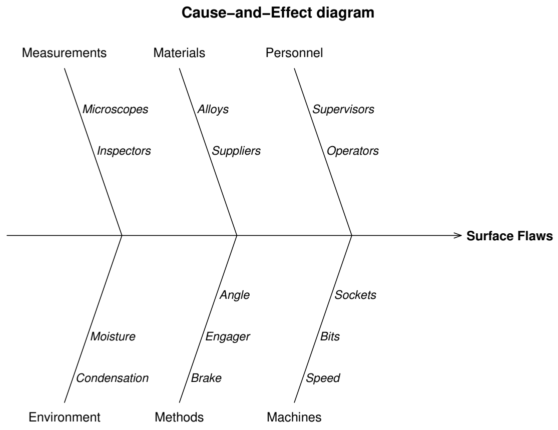

Figure 8: Cause-and-effect diagram.

Tuning and extending the package

Options to control some aspects of the **qcc** package can be set using the function `qcc.options`. If this is called with no arguments it retrieves the list of available options and their current values, otherwise it sets the required option. From a user point of view, the most interesting options allow to set the plotting character and the color used to highlight points beyond control limits or violating runs, the run length, i.e. the maximum value of a run before to signal a point as out of control, the background color used to draw the figure and the margin color, the character expansion used to draw the labels, the title and the statistics at the bottom of any plot. For details see `help(qcc.options)`.

Another interesting feature is the possibility to easily extend the package defining new control charts. Suppose we want to plot a p chart based on samples with different sizes. The resulting control chart will have non–constant control limits and, thus, it may result difficult to read by some operators. One possible solution is to draw a standardized control chart. This can be accomplished defining three new functions: one which computes and returns the group statistics to be charted (i.e. the z scores) and their center (zero in this case), another one which returns the within-group standard deviation (one in this case), and finally a function which returns the control limits. These functions need to be called stats.*type*, sd.*type* and limits.*type*, respectively, for a new chart of type "*type*". The following code may be used to define a standardized p chart:

```
stats.p.std <- function(data, sizes)
{
  data <- as.vector(data)
  sizes <- as.vector(sizes)
  pbar <- sum(data)/sum(sizes)
  z <- (data/sizes - pbar)/sqrt(pbar*(1-pbar)/sizes)
  list(statistics = z, center = 0)
}

sd.p.std <- function(data, sizes) return(1)

limits.p.std <- function(center, std.dev, sizes, conf)
{
  if (conf >= 1) { lcl <- -conf
                   ucl <- +conf }
  else
     { if (conf > 0 & conf < 1)
          { nsigmas <- qnorm(1 - (1 - conf)/2)
            lcl <- -nsigmas
            ucl <- +nsigmas }
       else stop("invalid 'conf' argument.") }
  limits <- matrix(c(lcl, ucl), ncol = 2)
  rownames(limits) <- rep("", length = nrow(limits))
  colnames(limits) <- c("LCL", "UCL")
  return(limits)
}
```

Then, we may source the above code and obtain the control charts in Figure 9 as follows:

```
# set unequal sample sizes
> n <- c(rep(50,5), rep(100,5), rep(25, 5))
# generate randomly the number of successes
> x <- rbinom(length(n), n, 0.2)
> par(mfrow=c(1,2))
# plot the control chart with variable limits
> qcc(x, type="p", size=n)
# plot the standardized control chart
> qcc(x, type="p.std", size=n)
```

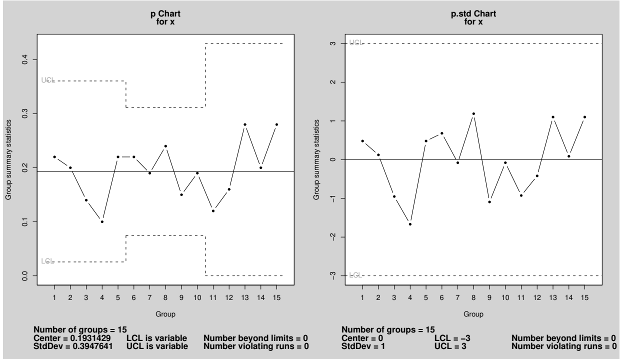

Figure 9: A p chart with non–constant control limits (left panel) and the corresponding standardized control chart (right panel).

Summary

In this paper we briefly describe the **qcc** package. This provides functions to draw basic Shewhart quality control charts for continuous, attribute and count data; corresponding operating characteristic curves are also implemented. Other statistical quality tools available are Cusum and EWMA charts for continuous data, process capability analyses, Pareto charts and cause-and-effect diagram.

References

Montgomery, D.C. (2000) *Introduction to Statistical Quality Control*, 4th ed. New York: John Wiley & Sons.
Wetherill, G.B. and Brown, D.W. (1991) *Statistical Process Control*. New York: Chapman & Hall.

Luca Scrucca
Dipartimento di Scienze Statistiche, Università degli Studi di Perugia, Italy
luca@stat.unipg.it

Least Squares Calculations in R

Timing different approaches

by Douglas Bates

Introduction

The S language encourages users to become programmers as they express new ideas and techniques of data analysis in S. Frequently the calculations in these techniques are expressed in terms of matrix operations. The subject of numerical linear algebra - how calculations with matrices can be carried out accurately and efficiently - is quite different from what many of us learn in linear algebra courses or in courses on linear statistical methods.

Numerical linear algebra is primarily based on decompositions of matrices, such as the LU decomposition, the QR decomposition, and the Cholesky decomposition, that are rarely discussed in linear algebra or statistics courses.

In this article we discuss one of the most common operations in statistical computing, calculating least squares estimates. We can write the problem mathematically as

$$\widehat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2 \qquad (1)$$

where X is an $n \times p$ model matrix ($p \leq n$), y is n-dimensional and β is p dimensional. Most statistics texts state that the solution to (1) is

$$\widehat{\beta} = \left(X'X\right)^{-1} X'y \qquad (2)$$

when X has full column rank (i.e. the columns of X are linearly independent).

Indeed (2) is a mathematically correct way of writing a least squares solution and, all too frequently, we see users of R calculating a least squares solution in exactly this way. That is, they use code like `solve(t(X) %*% X) %*% t(X) %*% y`. If the calculation is to be done only a few times on relatively small data sets then this is a reasonable way to do the calculation. However, it is rare that code only gets used only for small data sets or only a few times. By following a few rules we can make this code faster and more stable.

General principles

A few general principles of numerical linear algebra are:

1. Don't invert a matrix when you only need to solve a single system of equations. That is, use `solve(A, b)` instead of `solve(A) %*% b`.
2. Use `crossprod(X)`, not `t(X) %*% X` to calculate $X'X$. Similarly, use `crossprod(X,y)`, not `t(X) %*% y` to calculate $X'y$.
3. If you have a matrix with a special form, consider using a decomposition suited to that form. Because the matrix $X'X$ is symmetric and positive semidefinite, its Cholesky decomposition can be used to solve systems of equations.

Some timings on a large example

For a large, ill-conditioned least squares problem, such as that described in Koenker and Ng (2003), the literal translation of (2) does not perform well.

```
> library(Matrix)
> data(mm, y)
> mmm = as(mm, "matrix")
> dim(mmm)

[1] 1850  712

> sysgc.time(naive.sol <- solve(t(mmm) %*%
+     mmm) %*% t(mmm) %*% y)

[1] 3.64 0.18 4.11 0.00 0.00
```

(The function `sysgc.time` is a modified version of `system.time` that calls `gc()` before initializing the timer, thereby providing more consistent timing results. The data object `mm` is a sparse matrix, as described below, and we must coerce it to the dense matrix `mmm` to perform this calculation.)

According to the principles above, it should be more effective to compute

```
> sysgc.time(cpod.sol <- solve(crossprod(mmm),
+     crossprod(mmm, y)))

[1] 0.63 0.07 0.73 0.00 0.00

> all.equal(naive.sol, cpod.sol)

[1] TRUE
```

Timing results will vary between computers but in most cases the crossprod form of the calculation is at least four times as fast as the naive calculation. In fact, the entire crossprod solution is usually faster than calculating $X'X$ the naive way

```
> sysgc.time(t(mmm) %*% mmm)

[1] 0.83 0.03 0.86 0.00 0.00
```

because the `crossprod` function applied to a single matrix takes advantage of symmetry when calculating the product.

However, the value returned by `crossprod` does not retain the information that the product is symmetric (and positive semidefinite). As a result the solution of (1) is performed using a general linear system solver based on an LU decomposition when it would be faster, and more stable numerically, to use a Cholesky decomposition. The Cholesky decomposition could be used explicitly but the code is awkward

```
> sysgc.time(ch <- chol(crossprod(mmm)))

[1] 0.48 0.02 0.52 0.00 0.00

> sysgc.time(chol.sol <- backsolve(ch,
+     forwardsolve(ch, crossprod(mmm, y),
+      upper = TRUE, trans = TRUE)))

[1] 0.12 0.05 0.19 0.00 0.00

> all.equal(chol.sol, naive.sol)

[1] TRUE
```

Least squares calculations with Matrix classes

The Matrix package uses the S4 class system (Chambers, 1998) to retain information on the structure of matrices returned by intermediate calculations. A general matrix in dense storage, created by the Matrix function, has class "geMatrix". Its crossproduct has class "poMatrix". The solve methods for the "poMatrix" class use the Cholesky decomposition.

```
> mmg = as(mm, "geMatrix")
> class(crossprod(mmg))

[1] "poMatrix"
attr(,"package")
[1] "Matrix"

> sysgc.time(Mat.sol <- solve(crossprod(mmg),
+     crossprod(mmg, y)))

[1] 0.46 0.04 0.50 0.00 0.00

> all.equal(naive.sol, as(Mat.sol, "matrix"))

[1] TRUE
```

Furthermore, any method that calculates a decomposition or factorization stores the resulting factorization with the original object so that it can be reused without recalculation.

```
> xpx = crossprod(mmg)
> xpy = crossprod(mmg, y)
> sysgc.time(solve(xpx, xpy))

[1] 0.08 0.01 0.09 0.00 0.00

> sysgc.time(solve(xpx, xpy))

[1] 0.01 0.00 0.01 0.00 0.00
```

The model matrix mm is sparse; that is, most of the elements of mm are zero. The Matrix package incorporates special methods for sparse matrices, which produce the fastest results of all.

```
> class(mm)

[1] "cscMatrix"
attr(,"package")
[1] "Matrix"

> sysgc.time(sparse.sol <- solve(crossprod(mm),
+     crossprod(mm, y)))

[1] 0.03 0.00 0.04 0.00 0.00

> all.equal(naive.sol, as(sparse.sol, "matrix"))

[1] TRUE
```

The model matrix, mm, has class "cscMatrix" indicating that it is a compressed, sparse, column-oriented matrix. This is the usual representation for sparse matrices in the Matrix package. The class of crossprod(mm) is "sscMatrix" indicating that it is a symmetric, sparse, column-oriented matrix. The solve methods for such matrices first attempts to form a Cholesky decomposition. If this is successful the decomposition is retained as part of the object and can be reused in solving other systems based on this matrix. In this case, the decomposition is so fast that it is difficult to determine the difference in the solution times.

```
> xpx = crossprod(mm)
> class(xpx)

[1] "sscMatrix"
attr(,"package")
[1] "Matrix"

> xpy = crossprod(mm, y)
> sysgc.time(solve(xpx, xpy))

[1] 0.01 0.00 0.01 0.00 0.00

> sysgc.time(solve(xpx, xpy))

[1] 0.01 0.00 0.01 0.00 0.00
```

Epilogue

Another technique for speeding up linear algebra calculations is to use highly optimized libraries of Basic Linear Algebra Subroutines (BLAS) such as Kazushige Goto's BLAS (Goto and van de Geijn, 2002) or Atlas (Whaley et al., 2001). As described in R Development Core Team (2004), on many platforms R can be configured to use these enhanced BLAS libraries.

Most of the fast BLAS implementations focus on the memory access patterns within the calculations. In particular, they structure the calculation so that on-processor "cache" memory is used efficiently in basic operations such as dgemm for matrix-matrix multiplication.

Goto and van de Geijn (2002) describe how they calculate the matrix product $\boldsymbol{AB}$ by retrieving and storing parts of the transpose of $\boldsymbol{A}$. In the product it is the rows of $\boldsymbol{A}$ and the columns of $\boldsymbol{B}$ that will be accessed sequentially. In the Fortran column-major storage convention for matrices, which is what is used in R, elements in the same column are in adjacent storage locations. Thus the memory access patterns in the calculation $\boldsymbol{A}'\boldsymbol{B}$ are generally faster than those in $\boldsymbol{AB}$.

Notice that this means that calculating $\boldsymbol{A}'\boldsymbol{B}$ in R as t(A) %*% B instead of crossprod(A, B) causes A to be transposed twice; once in the calculation of t(A) and a second time in the inner loop of the matrix product. The `crossprod` function does not do any transposition of matrices – yet another reason to use `crossprod`.

Bibliography

J. M. Chambers. *Programming with Data*. Springer, New York, 1998. ISBN 0-387-98503-4. 19

K. Goto and R. van de Geijn. On reducing TLB misses in matrix multiplication. Technical Report TR02-55, Department of Computer Sciences, U. of Texas at Austin, 2002. 19

R. Koenker and P. Ng. SparseM: A sparse matrix package for R. *J. of Statistical Software*, 8(6), 2003. 18

R Development Core Team. *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, 2004. ISBN 3-900051-02-X. 19

R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001. Also available as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (`www.netlib.org/lapack/lawns/lawn147.ps`). 19

D.M. Bates
U. of Wisconsin-Madison
bates@wisc.edu

Tools for interactively exploring R packages

by Jianhua Zhang and Robert Gentleman

Introduction

An R package has the required and perhaps additional subdirectories containing files and information that may be of interest to users. Although these files can always be explored through command line operations both within and outside R, interactive tools that allow users to view and or operate on these files would be appealing to many users. Here we report on some widgets of this form that have been developed using the package *tcltk*. The functions are called `pExplorer`, `eExplorer`, and `vExplorer` and they are intended for exploring packages, examples and vignettes, respectively. These tools were developed as part of the Bioconductor project and are available in the package *tkWidgets* from www.bioconductor.org. Automatic downloading and package maintenance is also available using the *reposTools* mechanisms, also from Bioconductor.

Description

Our current implementation is in the form of `tcl/tk` widgets and we are currently adding similar functionality to the *RGtk* package, also available from the Bioconductor website. Users must have a functioning implementation (and be configured for) `tcl/tk`. Once both *tkWidgets* and *widgetTools* have been loaded into the working session the user has full access to these interactive tools.

pExplorer

`pExplorer` allows users to explore the contents of an R package. The widget can be invoked by providing a package name, a path to the directory containing the package, and the names of files and or subdirectory to be excluded from showing by the widget. The default behavior is to open the first package in the library of locally installed R (`.libPaths()`) with subdirectories/files named "Meta" and "latex" excluded if nothing is provided by a user.

Figure 1 shows the widget invoked by typing `pExplorer("base")`.

The following tasks can be performed through the interface:

- Typing in a valid path to a local directory containing R packages and then press the Enter key will update the name of the package being explored and show the contents of the first R package in the path in the `Contents` list box.
- Clicking the dropdown button for package path and selecting a path from the resulting dropdown list by double clicking will achieve the same results as typing in a new path.
- Clicking the `Browse` button allows users to selecting a directory containing R packages to achieve the same results as typing in a new path.
- Clicking the `Select Package` button allows users to pick a new package to explore and the name and contents of the package being explored are updated.

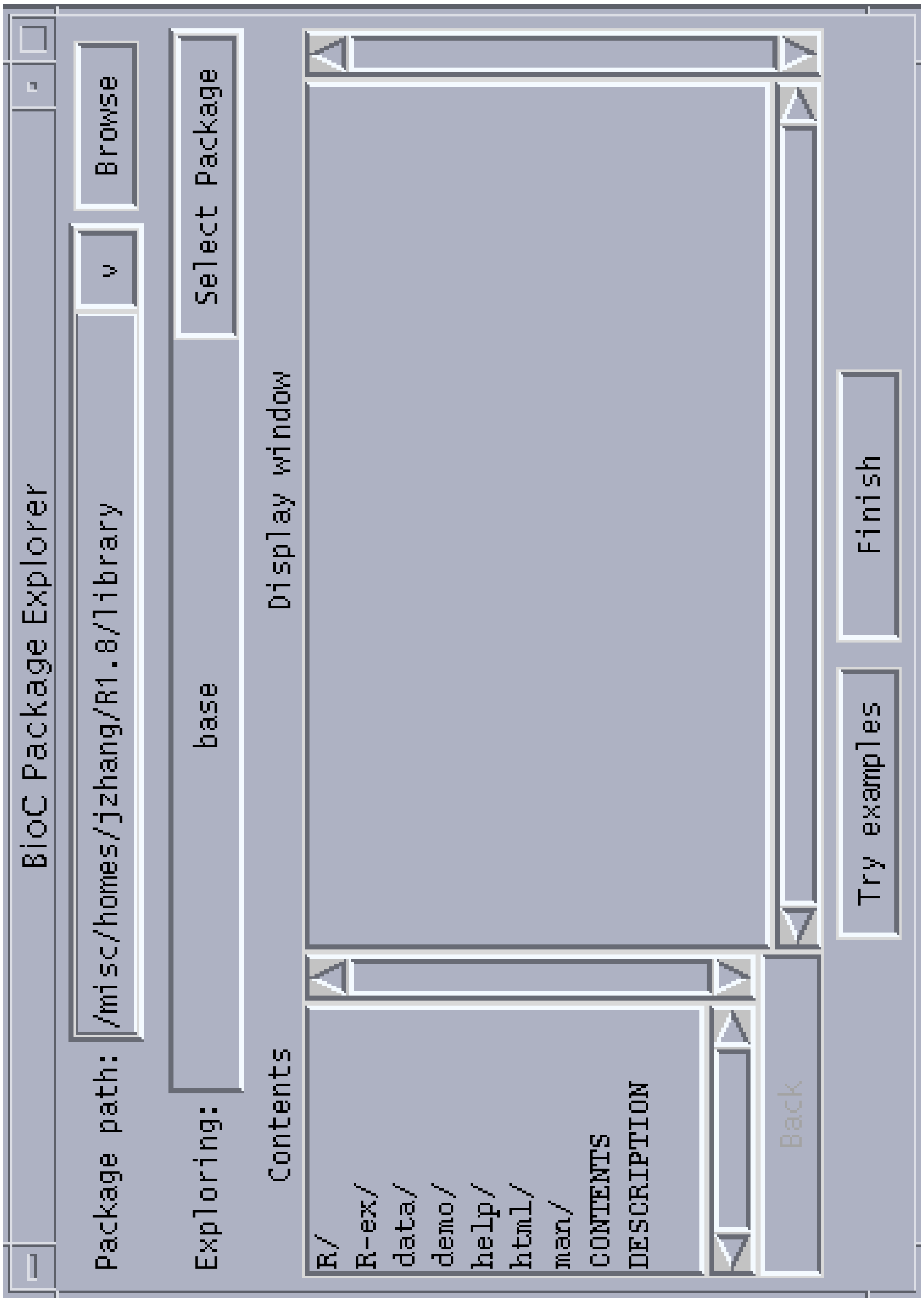

Figure 1: A snapshot of the widget when `pExplorer` is called

- Double clicking an item in the Contents list box results in different responses depending on the nature of the item being clicked. If the item is a subdirectory, the Contents will be updated with the contents of the subdirectory corresponding to the item just being clicked and the Back right below the list box will be activated. If the item is a file, the contents of the file will be displayed in the Display window text box.
- Clicking the Back button allows users to go back to the parent of the subdirectory whose contents are being displayed in Contents list box.
- Clicking the Try examples button invokes eExplorer going to be discussed next.

eExplorer

eExplorer allows users to explore the example code from all the help files for the specified package. When invoked by passing a valid package name, eExplorer will display the names of the files for example code stored in the R-ex subdirectory in the Example code chunk list box as shown by Figure 2.

Clicking an item in the list displays the code example in the R Source Code text box for users to view. Users may view the help file containing the example code by clicking the View Help button, execute the example code and have the output of the execution displayed in the Diaplay window by clicking the Execute Code button, or export the result of execution (if R objects have been created as the result of execution) to the global environment of the current R session by clicking the Export to R.

vExplorer

vExplorer allows users to explore the vignettes for a given R package by taking the advantages of the functionalities provided by Sweave (Leisch, 2002, 2003). When vExplorer is invoked without a package name as shown in Figure 3, the names of all the installed R packages containing vignettes will be shown in the Package List list box. When vExplorer is invoked with a package name, the Vignette List list box will be populated with the names of vignettes of the package.

Clicking a package name in the list box shows the names of all the vignettes of the package in the Vignette List list box. When a vignette name is clicked, a new widget as shown by Figure 4 appears that allows users to perform the following tasks if the selected vignettes contains any execute-able code chunks:

- Clicking an active button in the Code Chunk text box displays the code chunk in the R Source Code text box for users to view.
- Clicking the View PDF buttons shows the pdf version of the vignette currently being explored.
- Clicking the Execute Code button executes the code shown in the text box and displays the results of the execution in Results of Execution.
- Clicking the Export to R button export the result of execution to the global environment of the current R session.

The evaluation model for code chunks in a vignette is that they are evaluated sequentially. In order to ensure sequential evaluation and to help guide the user in terms of which code chunk is active we inactivate all buttons except the one associated with the first unevaluated code chunk.

References

Bibliography

Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002. URL http://CRAN.R-project.org/doc/Rnews/. 22

Friedrich Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, October 2003. URL http://CRAN.R-project.org/doc/Rnews/. 22

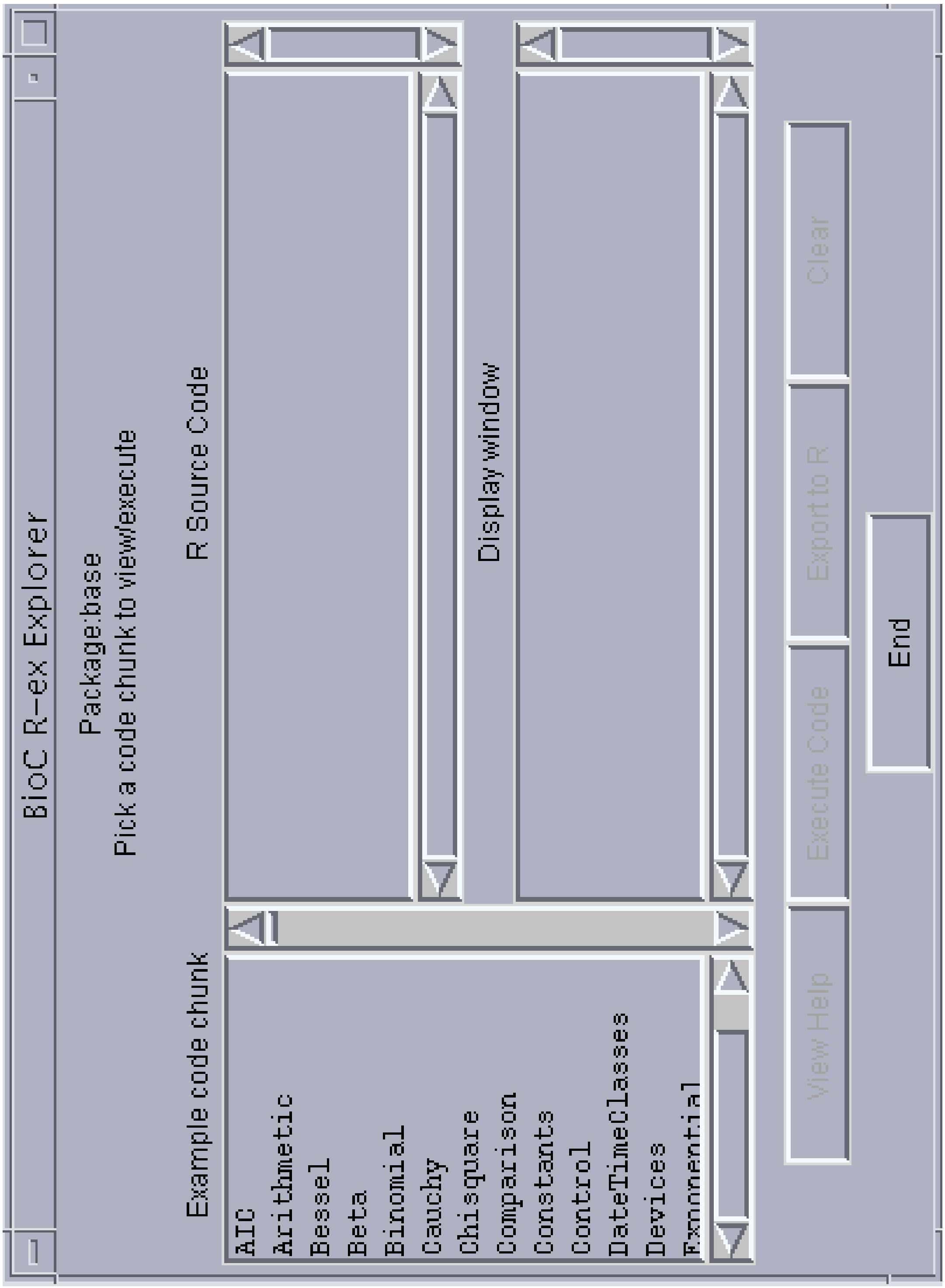

Figure 2: A snapshot of the widget when `pExplorer` is called by typing `eExplorer("base")`

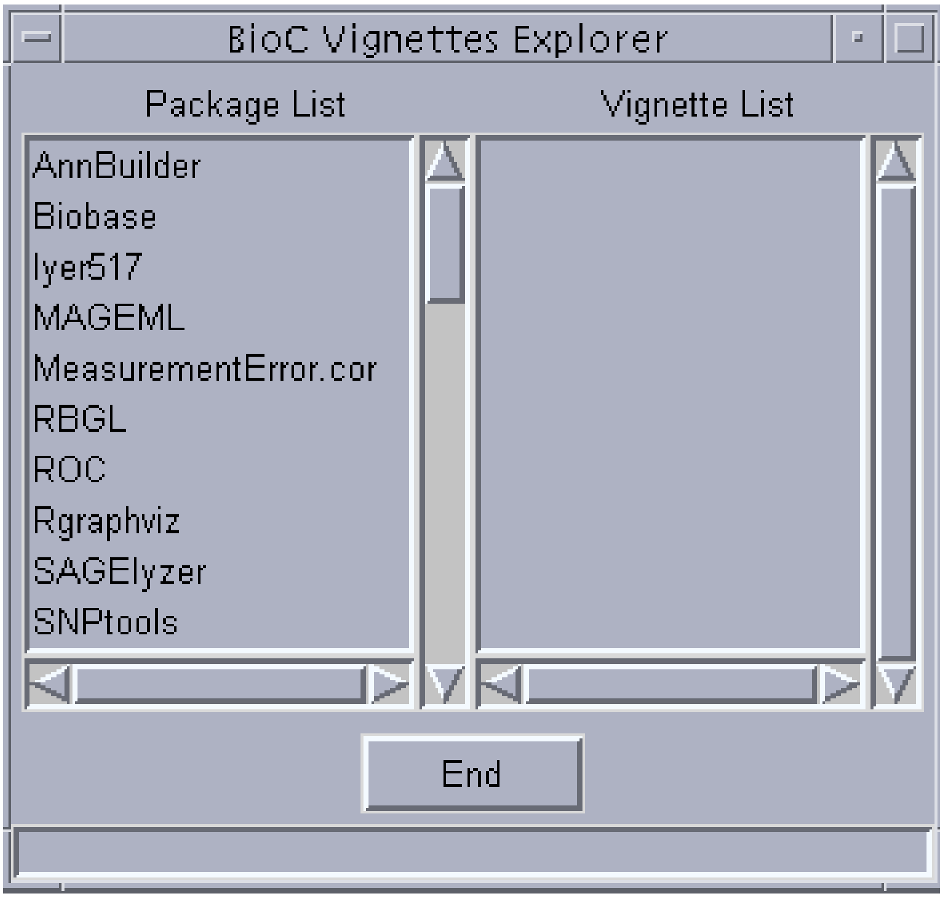

Figure 3: A snapshot of the widget when `vExplorer` is called by typing `vExplorer()`

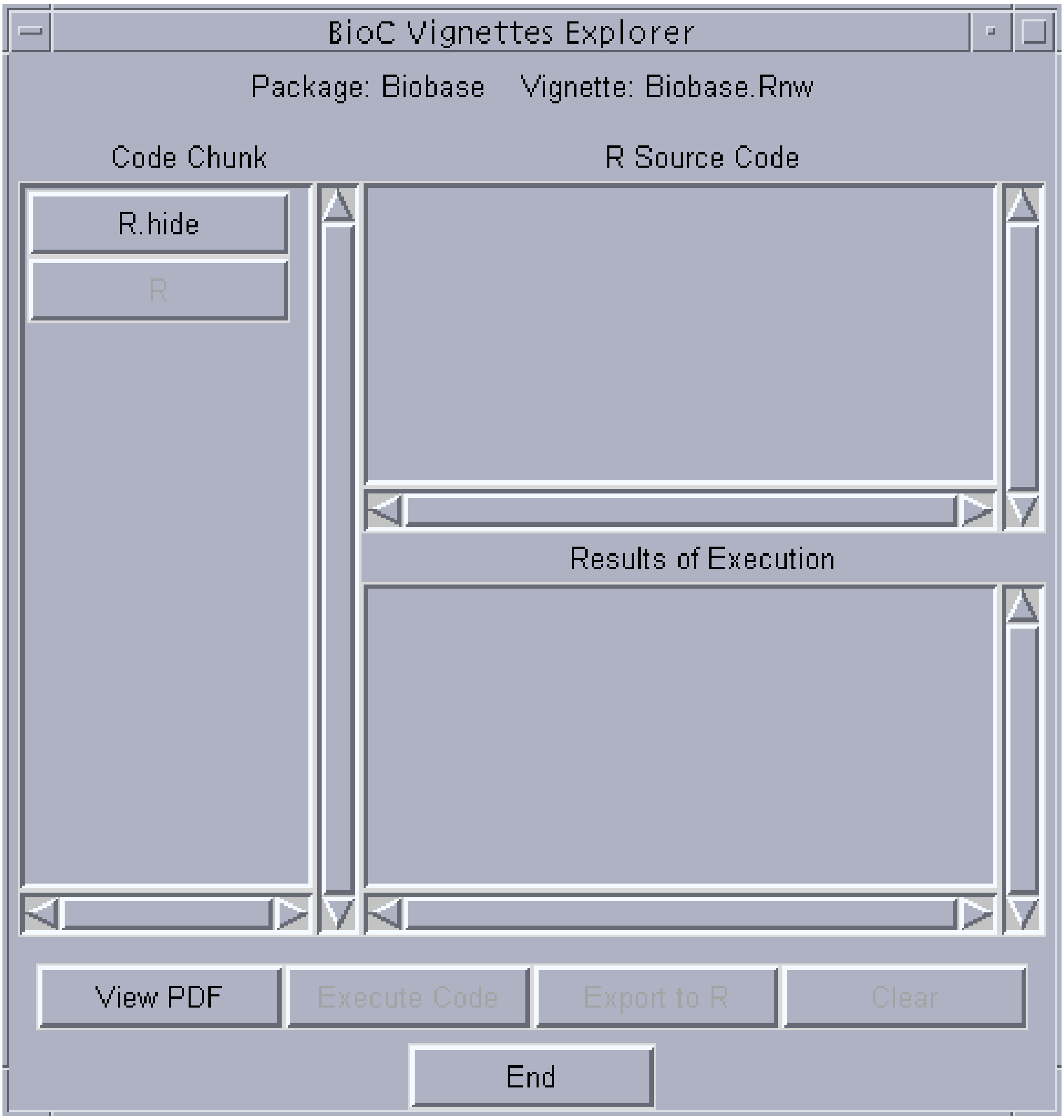

Figure 4: A snapshot of the widget when `Biobase.Rnw` of the *Biobase* package was clicked

The survival package

by Thomas Lumley

This is another in our series of articles describing the recommended packages that come with the R distribution. The survival package is a port of code written by T. M. Therneau, of the Mayo Clinic, code that has also been part of S-PLUS for many years. Porting the survival package in 1997 was my introduction to R programming. It exposed quite a few bugs in R and incompatibilities with S (some of which, on reflection, might have been better left incompatible).

Overview

Survival analysis, also called event history analysis in the social sciences and reliability analysis in engineering, is concerned with the time until an event occurs. The complication that makes survival analysis interesting is that individuals are often observed for only part of the relevant time period. They may be recruited well after time zero, and the study may end with only a small proportion of events having been seen.

In medical statistics the most common survival analyses are estimation of the distribution of time to the event, testing for equality of two distributions, and regression modelling of the rate at which events occur. These are all covered by the survival package.

Additional tools for regression modelling are in the Design and eha packages, and the muhaz package allows estimation of the hazard function (the analogue in survival analysis of density estimation).

Specifying survival data

In the popular 'counting process' formulation of survival analysis each record in a dataset has three variables describing the survival time. A `start` variable specifies when observation begins, a `stop` variable specifies when it ends, and an `event` variable is 1 if observation ended with an event and 0 if it ended without seeing the event. These variables are bundled into a `"Surv"` object by the `Surv()` function. If the `start` variable is zero for everyone it may be omitted.

In `data(veteran)`, `time` measures the time from the start of a lung cancer clinical trial and `status` is 1 if the patient died at `time`, 0 if follow-up ended with the patient still alive. In addition, `diagtime` gives the time from initial diagnosis until entry into the clinical trial.

```
> library(survival)
> data(veteran)
## time from randomisation to death
> with(veteran, Surv(time,status))
  [1]  72  411  228  126  118   10   82
  [8] 110  314  100+  42    8  144   25+
  ...

## time from diagnosis to death
> with(veteran, Surv(diagtime*30,
                     diagtime*30+time,
                     status))
  [1] ( 210, 282 ] ( 150, 561 ]
  [3] (  90, 318 ] ( 270, 396 ]
  ...
  [9] ( 540, 854 ] ( 180, 280+]
  ...
```

The first `"Surv"` object prints the observation time, with a + to indicate that the event is still to come. The second object prints the start and end of observation, again using a + to indicate end of follow-up before an event.

One and two sample summaries

The `survfit()` function estimates the survival distribution for a single group or multiple groups. It produces a `"survfit"` object with a `plot()` method. Figure 1 illustrates some of the options for producing more attractive graphs.

Two (or more) survival distributions can be compared with `survdiff`, which implements the logrank test (and the G^ρ family of weighted logrank tests). In this example it is clear from the graphs and the tests that the new treatment (`trt=2`) and the standard treatment (`trt=1`) were equally ineffective.

```
> data(veteran)
> plot(survfit(Surv(time,status)~trt,data=veteran),
     xlab="Years since randomisation",
     xscale=365, ylab="% surviving", yscale=100,
     col=c("forestgreen","blue"))

> survdiff(Surv(time,status)~trt, data=veteran)

Call:
survdiff(formula = Surv(time, status) ~ trt,
         data = veteran)

       N Observed Expected (O-E)^2/E
trt=1 69       64     64.5   0.00388
trt=2 68       64     63.5   0.00394
      (O-E)^2/V
trt=1   0.00823
trt=2   0.00823

 Chisq= 0  on 1 degrees of freedom, p= 0.928
```

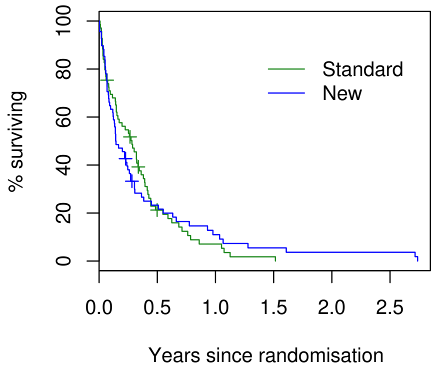

Figure 1: Survival distributions for two lung cancer treatments

Proportional hazards models

The mainstay of survival analysis in the medical world is the Cox proportional hazards model and its extensions. This expresses the hazard (or rate) of events as an unspecified baseline hazard function multiplied by a function of the predictor variables.

Writing $h(t; z)$ for the hazard at time t with predictor variables $Z = z$ the Cox model specifies

$$\log h(t, z) = \log h_0(t) e^{\beta z}.$$

Somewhat unusually for a semiparametric model, there is very little loss of efficiency by leaving $h_0(t)$ unspecified, and computation is, if anything, easier than for parametric models.

A standard example of the Cox model is one constructed at the Mayo Clinic to predict survival in patients with primary biliary cirrhosis, a rare liver disease. This disease is now treated by liver transplantation, but at the same there was no effective treatment. The model is based on data from 312 patients in a randomised trial.

```
> data(pbc)
> mayomodel<-coxph(Surv(time,status)~edtrt+
                    log(bili)+log(protime)+
                    age+platelet,
            data=pbc, subset=trt>0)
> mayomodel
Call:
coxph(formula = Surv(time, status) ~ edtrt +
          log(bili) + log(protime) +
           age + platelet, data = pbc,
           subset = trt > 0)


                 coef  exp(coef)
edtrt         1.02980      2.800
log(bili)     0.95100      2.588
log(protime)  2.88544     17.911
age           0.03544      1.036
platelet     -0.00128      0.999
             se(coef)      z       p
edtrt        0.300321   3.43 0.00061
log(bili)    0.097771   9.73 0.00000
log(protime) 1.031908   2.80 0.00520
age          0.008489   4.18 0.00003
platelet     0.000927  -1.38 0.17000

Likelihood ratio test=185  on 5 df, p=0  n= 312
```

The `survexp` function can be used to compare predictions from a proportional hazards model to actual survival. Here the comparison is for 106 patients who did not participate in the randomised trial. They are divided into two groups based on whether they had edema (fluid accumulation in tissues), an important risk factor.

```
> plot(survfit(Surv(time,status)~edtrt,
     data=pbc,subset=trt==-9))
> lines(survexp(~edtrt+
              ratetable(edtrt=edtrt,bili=bili,
                    platelet=platelet,age=age,
                    protime=protime),
              data=pbc,
              subset=trt==-9,
              ratetable=mayomodel,
              cohort=TRUE),
        col="purple")
```

The `ratetable` function in the model formula wraps the variables that are used to match the new sample to the old model.

Figure 2 shows the comparison of predicted survival (purple) and observed survival (black) in these 106 patients. The fit is quite good, especially as people who do and do not participate in a clinical trial are often quite different in many ways.

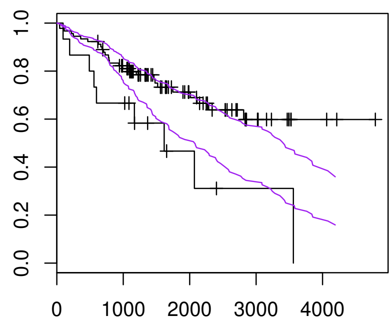

Figure 2: Observed and predicted survival

The main assumption of the proportional hazards model is that hazards for different groups are in fact proportional, *i.e.* that β is constant over time. The cox.zph function provides formal tests and graphical diagnostics for proportional hazards

```
> cox.zph(mayomodel)
                 rho chisq      p
edtrt        -0.1602 3.411 0.0648
log(bili)     0.1507 2.696 0.1006
log(protime) -0.1646 2.710 0.0997
age          -0.0708 0.542 0.4617
platelet     -0.0435 0.243 0.6221
GLOBAL            NA 9.850 0.0796

## graph for variable 1 (edtrt)
> plot(cox.zph(mayomodel)[1])
```

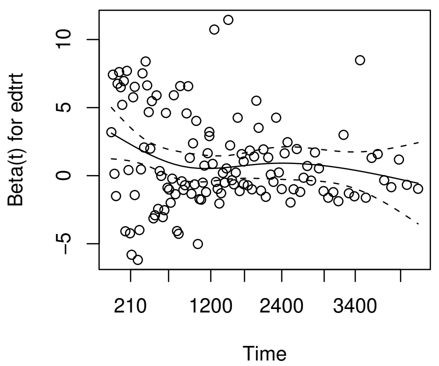

Figure 3: Testing proportional hazards

There is a suggestion that edtrt and log(protime) may have non-proportional hazards, and Figure 3 confirms this for edtrt. The curve shows an estimate of β at each point in time, obtained by smoothing the residuals on the graph. It appears that edtrt is quite strongly predictive for the first couple of years but has little predictive power after that.

Additional features

Computation for the Cox model extends straightforwardly to situations where z can vary over time, and when an individual can experience multiple events of the same or different types. Interpretation of the parameters becomes substantially trickier, of course. The coxph() function allows formulas to contain a cluster() term indicating which records belong to the same individual and a strata() term indicating subgroups that get their own unspecified baseline hazard function.

Complementing coxph is survreg, which fits linear models for the mean of survival time or its logarithm with various parametric error distributions. The parametric models allow more flexible forms of censoring than does the Cox model.

More recent additions include penalised likelihood estimation for smoothing splines and for random effects models.

The survival package also comes with standard mortality tables for the US and a few individual states, together with functions for comparing the survival of a sample to that of a population and computing person-years of followup.

Thomas Lumley
Department of Biostatistics
University of Washington, Seattle

useR! 2004

The R User Conference

by John Fox
McMaster University, Canada

More than 200 R users and developers converged on the Technische Universität Wien for the first R users' conference — userR! 2004 — which took place in Vienna between May 20 and May 22. The conference was organized by the Austrian Association for Statistical Computing (AASC) and sponsored by the R Foundation for Statistical Computing, the Austrian Science Foundation (FWF), and MedAnalytics (http://www.medanalytics.com/). Torsten Hothorn, Achim Zeileis, and David Meyer served as chairs of the conference, program committee, and local organizing committee; Bettina Grün was in charge of local arrangements.

The conference program included nearly 100 presentations, many in keynote, plenary, and semi-plenary sessions. The diversity of presentations — from bioinformatics to user interfaces, and finance to fights among lizards — reflects the wide range of applications supported by R.

A particularly interesting component of the program were keynote addresses by members of the R core team, aimed primarily at improving programming skills, and describing key features of R along with new and imminent developments. These talks reflected a major theme of the conference — the close relationship in R between use and development. The keynote addresses included:

- Paul Murrell on grid graphics and programming
- Martin Mächler on good programming practice

- Luke Tierney on namespaces and byte compilation
- Kurt Hornik on packaging, documentation, and testing
- Friedrich Leisch on S4 classes and methods
- Peter Dalgaard on language interfaces, using `.Call` and `.External`
- Douglas Bates on multilevel models in R
- Brian D. Ripley on datamining and related topics

Slides for the keynote addresses, along with abstracts of other papers presented at userR! 2004, are available at the conference web site, http://www.ci.tuwien.ac.at/Conferences/useR-2004/program.html.

An innovation of useR! 2004 was the deployment of "island" sessions, in place of the more traditional poster sessions, in which presenters with common interests made brief presentations followed by general discussion and demonstration. Several island sessions were conducted in parallel in the late afternoon on May 20 and 21, and conference participants were able to circulate among these sessions.

The conference sessions were augmented by a lively social program, including a pre-conference reception on the evening of May 19, and a conference trip, in the afternoon of May 22, to the picturesque Wachau valley on the Danube river. The trip was punctuated by a stop at the historic baroque Melk Abbey, and culminated in a memorable dinner at a traditional 'Heuriger' restaurant. Of course, much lively discussion took place at informal lunches, dinners, and pub sessions, and some of us took advantage of spring in Vienna to transform ourselves into tourists for a few days before and after the conference.

Everyone with whom I spoke, both at and after useR! 2004, was very enthusiastic about the conference. We look forward to an even larger useR! in 2006.

R Help Desk

Date and Time Classes in R

Gabor Grothendieck and Thomas Petzoldt

Introduction

R-1.9.0 and its contributed packages have a number of datetime (i.e. date or date/time) classes. In particular, the following three classes are discussed in this article:

- *Date*. This is the newest R date class, just introduced into R-1.9.0. It supports dates without times. Eliminating times simplifies dates substantially since not only are times, themselves, eliminated but the potential complications of time zones and daylight savings time vs. standard time need not be considered either. Date has an interface similar to the POSIX classes discussed below making it easy to move between them. It is part of the R base package. Dates in the Date class are represented internally as days since January 1, 1970. More information on Date can be found in `?Dates`. The percent codes used in character conversion with Date class objects can be found in `strptime` (although `strptime` itself is not a Date method).

- *chron*. Package **chron** provides dates and times. There are no time zones or notion of daylight vs. standard time in chron which makes it simpler to use for most purposes than date/time packages that do employ time zones. It is a contributed package available on CRAN so it must be installed and loaded prior to use. Datetimes in the **chron** package are represented internally as days since January 1, 1970 with times represented by fractions of days. Thus 1.5 would be the internal representation of noon on January 2nd, 1970. The **chron** package was developed at Bell Labs and is discussed in James and Pregibon (1993).

- *POSIX classes*. POSIX classes refer to the two classes POSIXct, POSIXlt and their common super class POSIXt. These support times and dates including time zones and standard vs. daylight savings time. They are primarily useful for time stamps on operating system files, data bases and other applications that involve external communication with systems that also support dates, times, time zones and daylight/standard times. They are also useful for certain applications such as world currency markets where time zones are important. We shall refer to these classes collectively as the POSIX classes. POSIXct datetimes are represented as seconds since January 1, 1970 GMT while POSIXlt datetimes are represented by a list of 9 components plus an optional tzone attribute. POSIXt is a common superclass of the two which is normally not accessed directly by the user. We shall focus mostly on POSIXct here. More information on the POSIXt classes are found in `?DateTimeClasses` and in `?strptime`. Additional insight can be obtained by inspecting the internals of POSIX datetime objects using `unclass(x)` where `x` is of any POSIX datetime class. (This works for Date and chron too). The POSIX classes are discussed in Ripley and Hornik (2001).

When considering which class to use, always choose the least complex class that will support the application. That is, *use Date if possible, otherwise use chron and otherwise use the POSIX classes*. Such a strategy will greatly reduce the potential for error and increase the reliability of your application.

A full page table is provided at the end of this article which illustrates and compares idioms written in each of the three datetime systems above.

Aside from the three primary date classes discussed so far there is also the **date** package, which represents dates as days since January 1, 1960. There are some date functions in the **pastecs** package that represent datetimes as years plus fraction of a year. The **date** and **pastecs** packages are not discussed in this article.

Other Applications

Spreadsheets like Microsoft Excel on a Windows PC or OpenOffice.org represent datetimes as days and fraction of days since December 30, 1899 (usually). If `x` is a vector of such numbers then `as.Date("1899-12-30") + floor(x)` will give a vector of Date class dates with respect to Date's origin. Similarly `chron("12/30/1899") + x` will give chron dates relative to chron's origin. Excel on a Mac usually represents dates as days and fraction of days since January 1, 1904 so `as.Date("1904-01-01") + floor(x)` and `chron("01/01/1904") + x` convert vectors of numbers representing such dates to Date and chron respectively. Its possible to set Excel to use either origin which is why the word *usually* was employed above.

SPSS uses October 14, 1582 as the origin thereby representing datetimes as seconds since the beginning of the Gregorian calendar. SAS uses seconds since January 1, 1960. `spss.get` and `sas.get` in package **Hmisc** can handle such datetimes automatically (Alzola and Harrell, 2002).

Time Series

A common use of dates and datetimes are in time series. The ts class represents regular time series (i.e. equally spaced points) and is mostly used if the frequency is monthly, quarterly or yearly. (Other series are labelled numerically rather than using dates.) Irregular time series can be handled by classes irts (in package **tseries**), its (in package **its**) and zoo (in package **zoo**). ts uses a scheme of its own for dates. irts and its use POSIXct dates to represent datetimes. zoo can use any date or timedate package to represent datetimes.

Input

By default, `read.table` will read in numeric data, such as `20040115`, as numbers and will read non-numeric data, such as `12/15/04` or `2004-12-15`, as factors. In either case, one should convert the data to character class using `as.character`. In the second (non-numeric) case one could alternately use the `as.is=` argument to `read.table` to prevent the conversion from character to factor within `read.table`[1].

```
# date col in all numeric format yyyymmdd
df <- read.table("laketemp.txt", header = TRUE)
as.Date(as.character(df$date), "%Y-%m-%d")

# first two cols in format mm/dd/yy hh:mm:ss
# Note as.is= in read.table to force character
library("chron")
df <- read.table("oxygen.txt", header = TRUE,
                 as.is = 1:2)
chron(df$date, df$time)
```

Avoiding Errors

The easiest way to avoid errors is to use the least complex class consistent with your application, as discussed in the Introduction.

With chron, in order to avoid conflicts between multiple applications, it is recommended that the user does not change the default settings of the four chron options:

```
# default origin
options(chron.origin=c(month=1,day=1,year=1970))
# if TRUE abbreviates year to 2 digits
options(chron.year.abb = TRUE)
# function to map 2 digit year to 4 digits
options(chron.year.expand = year.expand)
# if TRUE then year not displayed
options(chron.simplify = FALSE)
```

For the same reason, not only should the global chron origin not be changed but the per-variable origin should not be changed either. If one has a numeric vector of data x representing days since chron date `orig` then `orig+x` represents that data as chron dates relative to the default origin. With such a simple conversion there is really no reason to have to resort to non-standard origins. For example,

[1] This example uses data sets found at `http://www.tu-dresden.de/fghhihb/petzoldt/modlim/data/`.

```
orig <- chron("01/01/60")
x <- 0:9    # days since 01/01/60
# chron dates with default origin
orig + x
```

Regarding POSIX classes, the user should be aware of the following:

- *time zone*. Know which time zone each function that is being passed POSIX dates is assuming. Note that the time of day, day of the week, the day of the month, the month of the year, the year and other date quantities can potentially all differ depending on the time zone and whether it is daylight or standard time. The user *must* keep track of which time zone each function that is being used is assuming. For example, consider:

  ```
  dp <- seq(Sys.time(), len=10, by="day")
  plot(dp, 1:10)
  ```

  This does not use the current wall clock time for plotting today and the next 9 days since `plot` treats the datetimes as relative to GMT. The x values that result will be off from the wall clock time by a number of hours equal to the difference between the current time zone and GMT. See the plot example in table of this article for an illustration of how to handle this. Some functions accept a `tz=` argument, allowing the user to specify the time zone explicitly, but there are cases where the `tz=` argument is ignored. Always check the function with `tz=""` and `tz="GMT"` to be sure that `tz=` is not being ignored. If there is no `tz=` argument check carefully which time zone the function is assuming.

- *OS*. Some of the date calculations done by POSIX functions are passed off to the operating system and so may not work if the operating system has bugs with datetimes. In some cases the R code has compensated for OS bugs but in general *caveat emptor*. Another consequence is that different operating systems will accept different time zones. Normally the current time zone `""` and Greenwich Mean Time `"GMT"` can be assumed to be available but other time zones cannot be assumed to be available across platforms.

- *POSIXlt*. The `tzone` attribute on POSIXlt times are ignored so it is safer to use POSIXct than POSIXlt when performing arithmetic or other manipulations that may depend on time zones. Also, POSIXlt datetimes have an `isdst` component that indicates whether it is daylight savings time (`isdst=1`) or not (`isdst=0`). When converting from another class `isdst` may not be set as expected. Converting to character first and then to POSIXlt is safer than a direct conversion. Datetimes which are one hour off are the typical symptom of this last problem. Note that POSIXlt should not be used in data frames–POSIXct is preferred.

- *conversion*. Conversion between different datetime classes and POSIX classes may not use the time zone expected. Convert to character first to be sure. The conversion recipes given in the accompanying table should work as shown.

Comparison Table

Table 1 provides idioms written in each of the three classes discussed. The reader can use this table to quickly access the commands for those phrases in each of the three classes and to translate commands among the classes.

Bibliography

Alzola, C. F. and Harrell, F. E. (2002): *An Introduction to S and the Hmisc and Design Libraries*. URL `http://biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf`. 30

James, D. A. and Pregibon, D. (1993): Chronological objects for data analysis. In: *Proceedings of the 25th Symposium of the Interface*. San Diego. URL `http://cm.bell-labs.com/cm/ms/departments/sia/dj/papers/chron.pdf`. 29

Ripley, B. D. and Hornik, K. (2001): Date-time classes. *R News*, 1 (2), 8–11. URL `http://CRAN.R-project.org/doc/Rnews/`. 30

Gabor Grothendieck
ggrothendieck@myway.com

Thomas Petzoldt
petzoldt@rcs.urz.tu-dresden.de

	Date	chron	POSIXct
now	`Sys.Date()`	`as.chron(as.POSIXct(format(Sys.time()),tz="GMT"))`	`Sys.time()`
origin	`structure(0,class="Date")`	`chron(0)`	`structure(0,class=c("POSIXt","POSIXct"))`
x days since origin	`structure(floor(x),class="Date")`	`chron(x)`	`structure(x*24*60*60,class=c("POSIXt","POSIXct"))` # GMT days
diff in days	`dt1-dt2`	`dc1-dc2`	`difftime(dp1,dp2,unit="day")`[1]
time zone difference			`dp-as.POSIXct(format(dp,tz="GMT"))`
compare	`dt1 > dt2`	`dc1 > dc2`	`dp1 > dp2`
next day	`dt+1`	`dc+1`	`seq(dp0,length=2,by="DSTday")[2]`[2]
previous day	`dt-1`	`dc-1`	`seq(dp0,length=2,by="-1 DSTday")[2]`[3]
x days since date	`dt0+floor(x)`	dc0+x	`seq(dp0,length=2,by=paste(x,"DSTday")[2]`[4]
sequence	`dts <- seq(dt0,length=10,by="day")`	`dcs <- seq(dc0,length=10)`	`dps <- seq(dp0,length=10,by="DSTday")`
plot	`plot(dts,rnorm(10))`	`plot(dcs,rnorm(10))`	`plot(as.POSIXct(format(dps),tz="GMT"),rnorm(10))`[5]
every 2nd week	`seq(dt0,length=3,by="2 week")`	`dc0+seq(0,length=3,by=14)`	`seq(dp0,length=3,by="2 week")`
first day of month	`as.Date(format(dt,"%Y-%m-01"))`	`chron(dc)-month.day.year(dc)$day+1`	`as.POSIXct(format(dp,"%Y-%m-01"))`
month which sorts	`as.numeric(format(dt,"%m"))`	`months(dc)`	as.numeric(format(dp,"%m"))
day of week (Sun=0)	`as.numeric(format(dt,"%w"))`	`as.numeric(dates(dc)-3)%%7`	`as.numeric(format(dp,"%w"))`
day of year	`as.numeric(format(dt,"%j"))`	`as.numeric(format(as.Date(dc),"%j"))`	`as.numeric(format(dp,"%j"))`
mean each month/year	`tapply(x,format(dt,"%Y-%m"),mean)`	`tapply(x,format(as.Date(dc),"%Y-%m"),mean)`	`tapply(x,format(dp,"%Y-%m"),mean)`
12 monthly means	`tapply(x,format(dt,"%m"),mean)`	`tapply(x,months(dc),mean)`	`tapply(x,format(dp,"%m"),mean)`
Output			
yyyy-mm-dd	`format(dt)`	`format(as.Date(dates(dc)))`	`format(dp,"%Y-%m-%d")`
mm/dd/yy	`format(dt,"%m/%d/%y")`	`format(dates(dc))`	`format(dp,"%m/%d/%y")`
Sat Jul 1, 1970	`format(dt,"%a %b %d, %Y")`	`format(as.Date(dates(dc)),"%a %b %d, %Y")`	`format(dp,"%a %b %d, %Y")`
Input			
"1970-10-15"	`as.Date(z)`	`chron(z, format="y-m-d")`	`as.POSIXct(z)`
"10/15/1970"	`as.Date(z,"%m/%d/%Y")`	`chron(z)`	`as.POSIXct(strptime(z,"%m/%d/%Y"))`
Conversion (tz="")			
to Date		`as.Date(dates(dc))`	`as.Date(format(dp))`
to chron	`chron(unclass(dt))`		`chron(format(dp,"%m/%d/%Y"),format(dp,"%H:%M:%S"))`[6]
to POSIXct	`as.POSIXct(format(dt))`	`as.POSIXct(paste(as.Date(dates(dc)),times(dc)%%1))`	
Conversion (tz="GMT")			
to Date		`as.Date(dates(dc))`	`as.Date(dp)`
to chron	`chron(unclass(dt))`		`as.chron(dp)`
to POSIXct	`as.POSIXct(format(dt),tz="GMT")`	`as.POSIXct(dc)`	

Notes: z is a vector of characters. x is a vector of numbers. Variables beginning with dt are Date variables, Variables beginning with dc are chron variables and variables beginning with dp are POSIXct variables Variables ending in 0 are scalars; others are vectors. Expressions involving chron dates assume all chron options are set at default values. See strptime for more % codes

[1] Can be reduced to `dp1-dp2` if its acceptable that datetimes closer than one day are returned in hours or other units rather than days.
[2] Can be reduced to `dp+24*60*60` if one hour deviation at time changes between standard and daylight savings time are acceptable.
[3] Can be reduced to `dp-24*60*60` if one hour deviation at time changes between standard and daylight savings time are acceptable.
[4] Can be reduced to `dp+24*60*60*x` if one hour deviation acceptable when straddling odd number of standard/daylight savings time changes.
[5] Can be reduced to `plot(dps,rnorm(10))` if plotting points at GMT datetimes rather than current datetime suffices.
[6] An equivalent alternative is `as.chron(as.POSIXct(format(dp),tz="GMT"))`

Table 1: Comparison Table

Programmers' Niche

A simple class, in S3 and S4

by Thomas Lumley

ROC curves

Receiver Operating Characteristic curves (ROC curves) display the ability of an ordinal variable to discriminate between two groups. Invented by radio engineers, they are perhaps most used today in describing medical diagnostic tests. Suppose T is the result of a diagnostic test, and D is an indicator variable for the presence of a disease.

The ROC curve plots the true positive rate (sensitivity), $\Pr(T > c|D = 1)$, against the false positive rate (1−specificity), $\Pr(T > c|D = 0)$ for all values of c. Because the probabilities are conditional on disease status the ROC curve can be estimated from either an ordinary prospective sample or from separate samples of cases ($D = 1$) or controls ($D = 0$).

I will start out with a simple function to draw ROC curves, improve it, and then use it as the basis of S3 and S4 classes.

Simply coding up the definition of the ROC curve gives a function that is efficient enough for most practical pruposes. The one necessary optimisation is to realise that $\pm\infty$ and the the unique values of T are the only cutpoints needed. Note the use of `sapply` to avoid introducing an index variable for elements of `cutpoints`.

```
drawROC <-function(T,D){
  cutpoints<-c(-Inf, sort(unique(T)), Inf)
  sens<-sapply(cutpoints,
      function(c) sum(D[T>c])/sum(D))
  spec<-sapply(cutpoints,
      function(c) sum((1-D)[T<=c]/sum(1-D)))

  plot(1-spec, sens, type="l")
}
```

It would usually be bad style to use `T` and `c` as variable names, because of confusion with the S-compatible `T==TRUE` and the built-in function `c()`. In this case the benefit of preserving standard notation is arguably worth the potential confusion.

There is a relatively simple optimisation of the function that increases the speed substantially, though at the cost of requiring T to be a number, rather than just an object for which > and <= are defined.

```
drawROC<-function(T,D){
  DD<-table(-T,D)

  sens<-cumsum(DD[,2])/sum(DD[,2])
  mspec<-cumsum(DD[,1])/sum(DD[,1])
  plot( mspec,sens, type="l")
}
```

One pedagogical virtue of this code is that it makes it obvious that the ROC curve must be monotone: the true positive and false positive rates are cumulative sums of non-negative numbers.

Classes and methods

Creating an S3 class for ROC curves is an easy incremental step. The computational and graphical parts of `drawROC` are separated, and an object of class "ROC" is created simply by setting the `class` attribute of the result.

The first thing a new class needs is a `print` method. The `print` function is *generic*, when given an object of class "ROC" it automatically looks for a `print.ROC` function to take care of the work. Failing that, it calls `print.default`, which spews the internal representation of the object all over the screen.

```
ROC<-function(T,D){
   TT<-rev(sort(unique(T)))
   DD<-table(-T,D)

  sens<-cumsum(DD[,2])/sum(DD[,2])
  mspec<-cumsum(DD[,1])/sum(DD[,1])

  rval<-list(sens=sens, mspec=mspec, test=TT,
             call=sys.call())
  class(rval)<-"ROC"
  rval
}

print.ROC<-function(x,...){
  cat("ROC curve: ")
  print(x$call)
}
```

The programmer is responsible for ensuring that the object has all the properties assumed by functions that use ROC objects. If an object has class "duck", R will assume it can `look.duck`, `walk.duck` and `quack.duck`.

Since the main purpose of ROC curves is to be graphed, a `plot` method is also needed. As with `print`, `plot` will look for a `plot.ROC` method when handed an object of class "ROC" to plot.

```
plot.ROC<-function(x, type="b", null.line=TRUE,
     xlab="1-Specificity", ylab="Sensitivity",
     main=NULL, ...){
  par(pty="s")
  plot(x$mspec, x$sens, type=type,
         xlab=xlab, ylab=ylab, ...)
```

```
  if(null.line)
    abline(0, 1, lty=3)
  if(is.null(main))
    main<-x$call
  title(main=main)
}
```

A lines method allows ROC curves to be graphed on the same plot and compared. It is a stripped-down version of the plot method:

```
lines.ROC<-function(x,...){
  lines(x$mspec, x$sens, ...)
}
```

A method for identify allows the cutpoint to be found for any point on the curve:

```
identify.ROC<-function(x, labels=NULL,
                       ...,digits=1)
{
  if (is.null(labels))
    labels<-round(x$test,digits)
  identify(x$mspec, x$sens, labels=labels,...)
}
```

An AUC function, to compute the area under the ROC curve, is left as an exercise for the reader.

Generalising the class

The test variable *T* in an ROC curve may be a model prediction rather than a single biomarker, and ROC curves have been used to summarise the discriminatory power of logistic regression and survival models. A generic constructor function would allow methods for single variables and for models. Here the default method is the same as the original ROC function.

```
ROC <- function(T,...) UseMethod("ROC")
ROC.default<-function(T,D,...){
  TT<-rev(sort(unique(T)))
  DD<-table(-T,D)

  sens<-cumsum(DD[,2])/sum(DD[,2])
  mspec<-cumsum(DD[,1])/sum(DD[,1])

  rval<-list(sens=sens, mspec=mspec,
             test=TT,call=sys.call())
  class(rval)<-"ROC"
  rval
}
```

The ROC.glm method extracts the fitted values from a binomial regression model and uses them as the test variable.

```
ROC.glm<-function(T,...){
  if (!(T$family$family %in%
            c("binomial", "quasibinomial")))
    stop("ROC curves for binomial glms only")

  test<-fitted(T)
  disease<-(test+resid(T, type="response"))
  disease<-disease*weights(T)
  if (max(abs(disease %% 1))>0.01)
    warning("Y values suspiciously
                    far from integers")

  TT<-rev(sort(unique(test)))
  DD<-table(-test,disease)

  sens<-cumsum(DD[,2])/sum(DD[,2])
  mspec<-cumsum(DD[,1])/sum(DD[,1])

  rval<-list(sens=sens, mspec=mspec,
             test=TT,call=sys.call())
  class(rval)<-"ROC"
  rval
}
```

S4 classes and method

In the new S4 class system provided by the "methods" package, classes and methods must be registered. This ensures that an object has the components required by its class, and avoids the ambiguities of the S3 system. For example, it is not possible to tell from the name that t.test.formula is a method for t.test while t.data.frame is a method for t (and t.test.cluster, in the "Design" package, is neither).

As a price for this additional clarity, the S4 system takes a little more planning, and can be clumsy when a class needs to have components that are only sometimes present.

The definition of the ROC class is very similar to the calls used to create the ROC objects in the S3 functions.

```
setClass("ROC",
  representation(sens="numeric",mspec="numeric",
                 test="numeric",call="call"),
  validity=function(object)  {
     length(object@sens)==length(object@mspec) &&
     length(object@sens)==length(object@test)
  }
)
```

The first argument to setClass gives the name of the new class. The second describes the components (slots) that contain the data. Optional arguments include a validity check. In this case the ROC curve contains three numeric vectors and a copy of the call that created it. The validity check makes sure that the lengths of the three vectors agree. It could also check that the vectors were appropriately ordered, or that the true and false positive rates were between 0 and 1.

In contrast to the S3 class system, the S4 system requires all creation of objects to be done by the new function. This checks that the correct components are present and runs any validity checks. Here is the translation of the `ROC` function.

```
ROC<-function(T,D){
  TT<-rev(sort(unique(T)))
  DD<-table(-T,D)

  sens<-cumsum(DD[,2])/sum(DD[,2])
  mspec<-cumsum(DD[,1])/sum(DD[,1])

  new("ROC", sens=sens, mspec=mspec,
             test=TT,call=sys.call())

}
```

Rather than `print`, S4 objects use `show` for display. Here is a translation of the `print` method from the S3 version of the code

```
setMethod("show","ROC",
function(object){
  cat("ROC curve: ")
  print(object@call)
})
```

The first argument of `setMethod` is the name of the generic function (`show`). The second is the *signature*, which specifies when the method should be called. The signature is a vector of class names, in this case a single name since the generic function `show` has only one argument. The third argument is the method itself. In this example it is an anonymous function; it could also be a named function. The notation `object@call` is used to access the slots defined in `setClass`, in the same way that `$` is used for list components. The `@` notation should be used only in methods, although this is not enforced in current versions of R.

The `plot` method shows some of the added flexibility of the S4 method system. The generic function `plot` has two arguments, and the chosen method can be based on the classes of either or both. In this case a method is needed for the case where the first argument is an ROC object and there is no second argument, so the signature of the method is `c("ROC","missing")`. In addition to the two arguments `x` and `y` of the generic function, the method has all the arguments needed to customize the plot, including a `...` argument for further graphical parameters.

```
setMethod("plot", c("ROC","missing"),
    function(x, y, type="b", null.line=TRUE,
            xlab="1-Specificity",
            ylab="Sensitivity",
            main=NULL, ...){
       par(pty="s")
       plot(x@mspec, x@sens, type=type,
          xlab=xlab, ylab=ylab, ...)
       if(null.line)
          abline(0,1,lty=3)
       if(is.null(main))
          main<-x@call
       title(main=main)
       }
)
```

Creating a method for `lines` appears to work the same way

```
setMethod("lines","ROC",
    function(x,...)
        lines(x@mspec, x@sens,...)
)
```

In fact, things are more complicated. There is no S4 generic function for `lines` (unlike `plot` and `show`). When an S4 method is set on a function that is not already an S4 generic, a generic function is created. If you knew that `lines` was not already an S4 generic it would be good style to include a specific call to `setGeneric`, to make clear what is happening. Looking at the function `lines` before

```
> lines
function (x, ...)
UseMethod("lines")
<environment: namespace:graphics>
```

and after the `setMethod` call

```
> lines
standardGeneric for "lines" defined from
 package "graphics"

function (x, ...)
standardGeneric("lines")
<environment: 0x2fc68a8>
Methods may be defined for arguments: x
```

shows what happens.

Note that the creation of this S4 generic does not affect the workings of S3 methods for `lines`. Calling `methods("lines")` will still list the S3 methods, and calling `getMethods("lines")` will list both S3 and S4 methods.

Adding a method for creating ROC curves from binomial generalised linear models provides an example of `setGeneric`. Calling `setGeneric` creates a generic `ROC` function and makes the existing function the default method. The code is the same as the S3 `ROC.glm` except that `new` is used to create the ROC object.

```
setGeneric("ROC")

setMethod("ROC",c("glm","missing"),
  function(T){
    if (!(T$family$family %in%
         c("binomial", "quasibinomial")))
       stop("ROC curves for binomial glms only")
```

```
    test<-fitted(T)
    disease<-(test+resid(T,type="response"))
    disease<-disease*weights(T)
    if (max(abs(disease %% 1))>0.01)
        warning("Y values suspiciously far
                          from integers")

    TT<-rev(sort(unique(test)))
    DD<-table(-test,disease)

    sens<-cumsum(DD[,2])/sum(DD[,2])
    mspec<-cumsum(DD[,1])/sum(DD[,1])

    new("ROC",sens=sens, mspec=mspec,
              test=TT,call=sys.call())
  }
)
```

Finally, a method for `identify` shows one additional feature of `setGeneric`. The `signature` argument to `setGeneric` specifies which arguments are permitted in the signature of a method and thus are used for method dispatch. Method dispatch for `identify` will be based only on the first argument, which saves having to specify a `"missing"` second argument in the method.

```
setGeneric("identify",signature=c("x"))

setMethod("identify", "ROC",
  function(x, labels=NULL,
                  ...,digits=1){
    if (is.null(labels))
       labels<-round(x@test, digits)
    identify(x@mspec, x@sens,
            labels=labels,...)
    }
)
```

Discussion

Creating a simple class and methods requires very similar code whether the S3 or S4 system is used, and a similar incremental design strategy is possible. The S3 and S4 method system can coexist peacefully, even when S4 methods need to be defined for a function that already has S3 methods.

This example has not used inheritance, where the S3 and S4 systems differ more dramatically. Judging from the available examples of S4 classes, inheritance seems most useful in defining data structures, rather than objects representing statistical calculations. This may be because inheritance extends a class by creating a special case, but statisticians more often extend a class by creating a more general case. Reusing code from, say, linear models in creating generalised linear models is more an example of delegation than inheritance. It is not that a generalised linear model "is" a linear model, more that it "has" a linear model (from the last iteration of iteratively reweighted least squares) associated with it.

Thomas Lumley
Department of Biostatistics
University of Washington, Seattle

Changes in R

by the R Core Team

New features in 1.9.1

- `as.Date()` now has a method for "POSIXlt" objects.
- `mean()` has a method for "difftime" objects and so `summary()` works for such objects.
- `legend()` has a new argument `pt.cex`.
- `plot.ts()` has more arguments, particularly `yax.flip`.
- `heatmap()` has a new `keep.dendro` argument.
- The default `barplot` method now handles vectors and 1-d arrays (e.g., obtained by `table()`) the same, and uses grey instead of heat color palettes in these cases. (Also fixes PR#6776.)
- `nls()` now looks for variables and functions in its formula in the environment of the formula before the search path, in the same way `lm()` etc look for variables in their formulae.

User-visible changes in 1.9.0

- Underscore `_` is now allowed in syntactically valid names, and `make.names()` no longer changes underscores. Very old code that makes use of underscore for assignment may now give confusing error messages.
- Package 'base' has been split into packages 'base', 'graphics', 'stats' and 'utils'. All four are loaded in a default installation, but the separation allows a 'lean and mean' version of R to be used for tasks such as building indices.

  Packages ctest, eda, modreg, mva, nls, stepfun and ts have been merged into stats, and lqs has been returned to MASS. In all cases a stub has been left that will issue a warning and ensure that the appropriate new home is loaded. All the time series datasets have been moved to package stats. `Sweave` has been moved to utils.

Package mle has been moved to stats4 which will become the central place for statistical S4 classes and methods distributed with base R. Package mle remains as a stub.

Users may notice that code in .Rprofile is run with only the new base loaded and so functions may now not be found. For example, `ps.options(horizontal = TRUE)` should be preceded by `library(graphics)` or called as `graphics::ps.options` or, better, set as a hook – see `?setHook`.

- There has been a concerted effort to speed up the startup of an R session: it now takes about 2/3rds of the time of 1.8.1.

- A warning is issued at startup in a UTF-8 locale, as currently R only supports single-byte encodings.

New features in 1.9.0

- `$`, `$<-`, `[[`, `[[<-` can be applied to environments. Only character arguments are allowed and no partial matching is done. The semantics are basically that of `get/assign` to the environment with `inherits=FALSE`.

- There are now `print()` and `[` methods for "acf" objects.

- `aov()` will now handle singular `Error()` models, with a warning.

- `arima()` allows models with no free parameters to be fitted (to find log-likelihood and AIC values, thanks to Rob Hyndman).

- `array()` and `matrix()` now allow 0-length 'data' arguments for compatibility with S.

- `as.data.frame()` now has a method for arrays.

- `as.matrix.data.frame()` now coerces an all-logical data frame to a logical matrix.

- New function `assignInNamespace()` paralleling `fixInNamespace`.

- There is a new function `contourLines()` to produce contour lines (but not draw anything). This makes the CRAN package clines (with its `clines()` function) redundant.

- `D()`, `deriv()`, etc now also differentiate `asin()`, `acos()`, `atan()`, (thanks to a contribution of Kasper Kristensen).

- The `package` argument to `data()` is no longer allowed to be a (unquoted) name and so can be a variable name or a quoted character string.

- There is a new class "Date" to represent dates (without times) plus many utility functions similar to those for date-times. See `?Date`.

- Deparsing (including using `dump()` and `dput()`) an integer vector now wraps it in `as.integer()` so it will be source()d correctly. (Related to PR#4361.)

- `.Deprecated()` has a new argument `package` which is used in the warning message for non-base packages.

- The `print()` method for "difftime" objects now handles arrays.

- `dir.create()` is now an internal function (rather than a call to mkdir) on Unix as well as on Windows. There is now an option to suppress warnings from mkdir, which may or may not have been wanted.

- `dist()` has a new method to calculate Minkowski distances.

- `expand.grid()` returns appropriate array dimensions and dimnames in the attribute `"out.attrs"`, and this is used by the `predict()` method for loess to return a suitable array.

- `factanal()`, `loess()` and `princomp()` now explicitly check for numerical inputs; they might have silently coded factor variables in formulae.

- New functions `factorial()` defined as `gamma(x+1)` and for S-PLUS compatibility, `lfactorial()` defined as `lgamma(x+1)`.

- `findInterval(x, v)` now allows +/-Inf values, and NAs in `x`.

- `formula.default()` now looks for a `"terms"` component before a `formula` argument in the saved call: the component will have '.' expanded and probably will have the original environment set as its environment. And what it does is now documented.

- `glm()` arguments `etastart` and `mustart` are now evaluated via the model frame in the same way as `subset` and `weights`.

- Functions grep(), regexpr(), sub() and gsub() now coerce their arguments to character, rather than give an error.

  The perl=TRUE argument now uses character tables prepared for the locale currently in use each time it is used, rather than those of the C locale.

- New functions head() and tail() in package 'utils'. (Based on a contribution by Patrick Burns.)

- legend() has a new argument 'text.col'.

- methods(class=) now checks for a matching generic, and so no longer returns methods for non-visible generics (and eliminates various mismatches).

- A new function mget() will retrieve multiple values from an environment.

- model.frame() methods, for example those for "lm" and "glm", pass relevant parts of ... onto the default method. (This has long been documented but not done.) The default method is now able to cope with model classes such as "lqs" and "ppr".

- nls() and ppr() have a model argument to allow the model frame to be returned as part of the fitted object.

- "POSIXct" objects can now have a "tzone" attribute that determines how they will be converted and printed. This means that date-time objects which have a timezone specified will generally be regarded as in their original time zone.

- postscript() device output has been modified to work around rounding errors in low-precision calculations in gs >= 8.11. (PR#5285, which is not a bug in R.)

  It is now documented how to use other Computer Modern fonts, for example italic rather than slanted.

- ppr() now fully supports categorical explanatory variables,

  ppr() is now interruptible at suitable places in the underlying FORTRAN code.

- princomp() now warns if both x and covmat are supplied, and returns scores only if the centring used is known.

- psigamma(x, deriv=0), a new function generalizes, digamma() etc. All these (psigamma, digamma, trigamma,...) now also work for x < 0.

- pchisq(, ncp > 0) and hence qchisq() now work with much higher values of ncp; it has become much more accurate in the left tail.

- read.table() now allows embedded newlines in quoted fields. (PR#4555)

- rep.default(0-length-vector, length.out=n) now gives a vector of length n and not length 0, for compatibility with S.

  If both each and length.out have been specified, it now recycles rather than fills with NAs for S compatibility.

  If both times and length.out have been specified, times is now ignored for S compatibility. (Previously padding with NAs was used.)

  The "POSIXct" and "POSIXlt" methods for rep() now pass ... on to the default method (as expected by PR#5818).

- rgb2hsv() is new, an R interface the C API function with the same name.

- User hooks can be set for onLoad, library, detach and onUnload of packages/namespaces: see ?setHook.

- save() default arguments can now be set using option "save.defaults", which is also used by save.image() if option "save.image.defaults" is not present.

- New function shQuote() to quote strings to be passed to OS shells.

- sink() now has a split= argument to direct output to both the sink and the current output connection.

- split.screen() now works for multiple devices at once.

- On some OSes (including Windows and those using glibc) strptime() did not validate dates correctly, so we have added extra code to do so. However, this cannot correct scanning errors in the OS's strptime (although we have been able to work around these on Windows). Some examples are now tested for during configuration.

- strsplit() now has fixed and perl arguments and split="" is optimized.

- subset() now allows a drop argument which is passed on to the indexing method for data frames.

- termplot() has an option to smooth the partial residuals.

- varimax() and promax() add class "loadings" to their loadings component.

- Model fits now add a `"dataClasses"` attribute to the terms, which can be used to check that the variables supplied for prediction are of the same type as those used for fitting. (It is currently used by `predict()` methods for classes `"lm"`, `"mlm"`, `"glm"` and `"ppr"`, as well as methods in packages MASS, rpart and tree.)

- New command-line argument `--max-ppsize` allows the size of the pointer protection stack to be set higher than the previous limit of 10000.

- The fonts on an `X11()` device (also `jpeg()` and `png()` on Unix) can be specified by a new argument 'fonts' defaulting to the value of a new option `"X11fonts"`.

- New functions in the tools package: `pkgDepends`, `getDepList` and `installFoundDepends`. These provide functionality for assessing dependencies and the availability of them (either locally or from online repositories).

- The parsed contents of a NAMESPACE file are now stored at installation and if available used to speed loading the package, so packages with namespaces should be reinstalled.

- Argument `asp`, although not a graphics parameter, is accepted in the `...` of graphics functions without a warning. It now works as expected in `contour()`.

- Package stats4 exports S4 generics for `AIC()` and `BIC()`.

- The Mac OS X version now produces an R framework for easier linking of R into other programs. As a result, R.app is now relocatable.

- Added experimental support for conditionals in NAMESPACE files.

- Added `as.list.environment` to coerce environments to lists (efficiently).

- New function `addmargins()` in the stats package to add marginal summaries to tables, e.g. row and column totals. (Based on a contribution by Bendix Carstensen.)

- dendrogam edge and node labels can now be expressions (to be plotted via `stats:::plotNode` called from `plot.dendrogram`). The diamond frames around edge labels are more nicely scaled horizontally.

- Methods defined in the methods package can now include default expressions for arguments. If these arguments are missing in the call, the defaults in the selected method will override a default in the generic. See `?setMethod`.

- Changes to package 'grid':

  - Renamed `push/pop.viewport()` to `push/popViewport()`.

  - Added `upViewport()`, `downViewport()`, and `seekViewport()` to allow creation and navigation of viewport tree (rather than just viewport stack).

  - Added `id` and `id.lengths arguments` to `grid.polygon()` to allow multiple polygons within single `grid.polygon()` call.

  - Added `vpList()`, `vpStack()`, `vpTree()`, and `current.vpTree()` to allow creation of viewport "bundles" that may be pushed at once (lists are pushed in parallel, stacks in series).

    `current.vpTree()` returns the current viewport tree.

  - Added `vpPath()` to allow specification of viewport path in `downViewport()` and `seekViewport()`.

    See `?viewports` for an example of its use.

    NOTE: it is also possible to specify a path directly, e.g., something like `vp1::vp2`, but this is only advised for interactive use (in case I decide to change the separator :: in later versions).

  - Added `just` argument to `grid.layout()` to allow justification of layout relative to parent viewport **IF** the layout is not the same size as the viewport. There's an example in help(grid.layout).

  - Allowed the `"vp"` slot in a grob to be a viewport name or a vpPath. The interpretation of these new alternatives is to call `downViewport()` with the name or vpPath before drawing the grob and `upViewport()` the appropriate amount after drawing the grob. Here's an example of the possible usage:

```
pushViewport(viewport(w=.5, h=.5,
                      name="A"))
grid.rect()
pushViewport(viewport(w=.5, h=.5,
                      name="B"))
grid.rect(gp=gpar(col="grey"))
upViewport(2)
grid.rect(vp="A",
          gp=gpar(fill="red"))
grid.rect(vp=vpPath("A", "B"),
          gp=gpar(fill="blue"))
```

  - Added `engine.display.list()` function. This allows the user to tell grid NOT to use the graphics engine display list and to handle ALL redraws using its own display list (including redraws after device resizes and copies).

    This provides a way to avoid some of the problems with resizing a device when you have used `grid.convert()`, or the gridBase package, or even base functions such as `legend()`.

    There is a document discussing the use of display lists in grid on the grid web site http://www.stat.auckland.ac.nz/~paul/grid/grid.html

  - Changed the implementation of grob objects. They are no longer implemented as external references. They are now regular R objects which copy-by-value. This means that they can be saved/loaded like normal R objects. In order to retain some existing grob behaviour, the following changes were necessary:

    * grobs all now have a `"name"` slot. The grob name is used to uniquely identify a `"drawn"` grob (i.e., a grob on the display list).
    * `grid.edit()` and `grid.pack()` now take a grob name as the firs argument instead of a grob. (Actually, they take a gPath; see below)
    * the `"grobwidth"` and `"grobheight"` units take either a grob OR a grob name (actually a gPath; see below). Only in the latter case will the unit be updated if the grob "pointed to" is modified.

    In addition, the following features are now possible with grobs:

    * grobs now `save()`/`load()` like any normal R object.
    * many `grid.*()` functions now have a `*Grob()` counterpart. The `grid.*()` version is used for its side-effect of drawing something or modifying something which has been drawn; the `*Grob()` version is used for its return value, which is a grob. This makes it more convenient to just work with grob objects without producing any graphical output (by using the `*Grob()` functions).
    * there is a `gTree object` (derived from grob), which is a grob that can have children. A gTree also has a `"childrenvp"` slot which is a viewport which is pushed and then "up"ed before the children are drawn; this allows the children of a `gTree` to place themselves somewhere in the viewports specified in the `childrenvp` by having a `vpPath` in their `vp` slot.
    * there is a `gPath` object, which is essentially a concatenation of grob names. This is used to specify the child of (a child of ...) a `gTree`.
    * there is a new API for creating/accessing/modifying grob objects: `grid.add()`, `grid.remove()`, `grid.edit()`, `grid.get()` (and their `*Grob()` counterparts can be used to add, remove, edit, or extract a grob or the child of a gTree. NOTE: the new `grid.edit()` API is incompatible with the previous version.

  - Added `stringWidth()`, `stringHeight()`, `grobWidth()`, and `grobHeight()` convenience functions (they produce "strwidth", "strheight", "grobwidth", and "grobheight" unit objects, respectively).

  - Allowed viewports to turn off clipping altogether. Possible settings for viewport clip arg are now:

    "on" clip to the viewport (was `TRUE`)

    "inherit" clip to whatever parent says (was `FALSE`)

    "off" turn off clipping

    Still accept logical values (and NA maps to "off")

- `R CMD check` now runs the (Rd) examples with default RNGkind (uniform & normal) and codeset.seed(1). `example(*, setRNG = TRUE)` does the same.

- `undoc()` in package 'tools' has a new default of 'use.values = NULL' which produces a warning whenever the default values of function arguments differ between documentation and code. Note that this affects `R CMD check` as well.

- Testing examples via `massage-examples.pl` (as used by `R CMD check`) now restores the search path after every help file.

- `checkS3methods()` in package 'tools' now also looks for generics in the loaded namespaces/packages listed in the Depends fields of the package's DESCRIPTION file when testing an installed package.

- The DESCRIPTION file of packages may contain a 'Suggests:' field for packages that are used only in examples or vignettes.

- Added an option to package.dependencies() to handle the 'Suggests' levels of dependencies.
- Vignette dependencies can now be checked and obtained via vignetteDepends.
- Option "repositories" to list URLs for package repositories added.
- package.description() has been replaced by packageDescription().
- R CMD INSTALL/build now skip Subversion's .svn directories as well as CVS directories.
- arraySubscript and vectorSubscript take a new argument which is a function pointer that provides access to character strings (such as the names vector) rather than assuming these are passed in.
- R_CheckUserInterrupt is now described in 'Writing R Extensions' and there is a new equivalent subroutine rchkusr for calling from FORTRAN code.
- hsv2rgb and rgb2hsv are newly in the C API.
- Salloc and Srealloc are provided in S.h as wrappers for S_alloc and S_realloc, since current S versions use these forms.
- The type used for vector lengths is now R_len_t rather than int, to allow for a future change.
- The internal header nmath/dpq.h has slightly improved macros R_DT_val() and R_DT_Cval(), a new R_D_LExp() and improved R_DT_log() and R_DT_Clog(); this improves accuracy in several [dpq]-functions for extreme arguments.
- print.coefmat() is defunct, replaced by printCoefmat().
- codes() and codes<-() are defunct.
- anovalist.lm (replaced in 1.2.0) is now defunct.
- glm.fit.null(), lm.fit.null() and lm.wfit.null() are defunct.
- print.atomic() is defunct.
- The command-line arguments --nsize and --vsize are no longer recognized as synonyms for --min-nsize and --min-vsize (which replaced them in 1.2.0).
- Unnecessary methods {coef.{g}lm and fitted.{g}lm have been removed: they were each identical to the default method.
- La.eigen() is deprecated now eigen() uses LAPACK by default.
- tetragamma() and pentagamma() are deprecated, since they are equivalent to psigamma(, deriv=2) and psigamma(, deriv=3).
- LTRUE/LFALSE in Rmath.h have been removed: they were deprecated in 1.2.0.
- package.contents() and package.description() have been deprecated.
- The defaults for configure are now --without-zlib --without-bzlib --without-pcre.

  The included PCRE sources have been updated to version 4.5 and PCRE >= 4.0 is now required if --with-pcre is used.

  The included zlib sources have been updated to 1.2.1, and this is now required if --with-zlib is used.
- configure no longer lists bzip2 and PCRE as 'additional capabilities' as all builds of R have had them since 1.7.0.
- --with-blasgoto= to use K. Goto's optimized BLAS will now work.

The above lists only new features, see the 'NEWS' file in the R distribution or on the R homepage for a list of bug fixes.

Changes on CRAN

by Kurt Hornik

New contributed packages

AlgDesign Algorithmic experimental designs. Calculates exact and approximate theory experimental designs for D, A, and I criteria. Very large designs may be created. Experimental designs may be blocked or blocked designs created from a candidate list, using several criteria. The blocking can be done when whole and within plot factors interact. By Bob Wheeler.

BradleyTerry Specify and fit the Bradley-Terry model and structured versions. By David Firth.

BsMD Bayes screening and model discrimination follow-up designs. By Ernesto Barrios based on Daniel Meyer's code.

DCluster A set of functions for the detection of spatial clusters of disease using count data. Bootstrap is used to estimate sampling distributions of statistics. By Virgilio Gómez-Rubio, Juan Ferrándiz, Antonio López.

GeneTS A package for analyzing multiple gene expression time series data. Currently, **GeneTS** implements methods for cell cycle analysis and for inferring large sparse graphical Gaussian models. For plotting the inferred genetic networks **GeneTS** requires the **graph** and **Rgraphviz** packages (available from `www.bioconductor.org`). By Konstantinos Fokianos, Juliane Schaefer, and Korbinian Strimmer.

HighProbability Provides a simple, fast, reliable solution to the multiple testing problem. Given a vector of p-values or achieved significance levels computed using standard frequentist inference, **HighProbability** determines which ones are low enough that their alternative hypotheses can be considered highly probable. The p-value vector may be determined using existing R functions such as `t.test`, `wilcox.test`, `cor.test`, or `sample`. **HighProbability** can be used to detect differential gene expression and to solve other problems involving a large number of hypothesis tests. By David R. Bickel.

Icens Many functions for computing the NPMLE for censored and truncated data. By R. Gentleman and Alain Vandal.

MCMCpack This package contains functions for posterior simulation for a number of statistical models. All simulation is done in compiled C++ written in the Scythe Statistical Library Version 0.4. All models return coda mcmc objects that can then be summarized using coda functions or the coda menu interface. The package also contains some useful utility functions, including some additional PDFs and pseudo-random number generators for statistical distributions. By Andrew D. Martin, and Kevin M. Quinn.

MNP A publicly available R package that fits the Bayesian Multinomial Probit models via Markov chain Monte Carlo. Along with the standard Multinomial Probit model, it can also fit models with different choice sets for each observation and complete or partial ordering of all the available alternatives. The estimation is based on the efficient marginal data augmentation algorithm that is developed by Imai and van Dyk (2004). By Kosuke Imai, Jordan Vance, David A. van Dyk.

NADA Contains methods described by Dennis R. Helsel in his book "Nondetects And Data Analysis: Statistics for Censored Environmental Data". By Lopaka Lee.

R2WinBUGS Using this package, it is possible to call a BUGS model, summarize inferences and convergence in a table and graph, and save the simulations in arrays for easy access in R. By originally written by Andrew Gelman; changes and packaged by Sibylle Sturtz and Uwe Ligges.

RScaLAPACK An interface to ScaLAPACK functions from R. By Nagiza F. Samatova, Srikanth Yoginath, and David Bauer.

RUnit R functions implementing a standard Unit Testing framework, with additional code inspection and report generation tools. By Matthias Burger, Klaus Juenemann, Thomas Koenig.

SIN This package provides routines to perform SIN model selection as described in Drton & Perlman (2004). The selected models are in the format of the **ggm** package, which allows in particular parameter estimation in the selected model. By Mathias Drton.

SoPhy SWMS_2D interface, Submission to Computers and Geosciences, title: The use of the language interface of R: two examples for modelling water flux and solute transport. By Martin Schlather, Bernd Huwe.

SparseLogReg Some functions wrapping the sparse logistic regression code by S. K. Shevade and S. S. Keerthi originally intended for microarray-based gene selection. By Michael T. Mader, with C code from S. K. Shevade and S. S. Keerthi.

assist ASSIST, see manual. By Yuedong Wang, and Chunlei Ke.

bayesmix Bayesian mixture models of univariate Gaussian distributions using JAGS. By Bettina Gruen.

betareg Beta regression for modeling rates and proportions. By Alexandre de Bustamante Simas.

circular Circular Statistics, from "Topics in circular Statistics" (2001) S. Rao Jammalamadaka and A. SenGupta, World Scientific. By Ulric Lund, Claudio Agostinelli.

crossdes Contains functions for the construction and randomization of balanced carryover balanced designs. Contains functions to check given designs for balance. Also contains functions for simulation studies on the validity of two randomization procedures. By Martin Oliver Sailer.

debug Debugger for R functions, with code display, graceful error recovery, line-numbered conditional breakpoints, access to exit code, flow control, and full keyboard input. By Mark V. Bravington.

distr Object orientated implementation of distributions and some additional functionality. By Florian Camphausen, Matthias Kohl, Peter Ruckdeschel, Thomas Stabla.

dynamicGraph Interactive graphical tool for manipulating graphs. By Jens Henrik Badsberg.

energy E-statistics (energy) tests for comparing distributions: multivariate normality, Poisson test, multivariate *k*-sample test for equal distributions, hierarchical clustering by e-distances. Energy-statistics concept based on a generalization of Newton's potential energy is due to Gabor J. Szekely. By Maria L. Rizzo and Gabor J. Szekely.

evdbayes Provides functions for the bayesian analysis of extreme value models, using MCMC methods. By Alec Stephenson.

evir Functions for extreme value theory, which may be divided into the following groups; exploratory data analysis, block maxima, peaks over thresholds (univariate and bivariate), point processes, gev/gpd distributions. By S original (EVIS) by Alexander McNeil, R port by Alec Stephenson.

fBasics fBasics package from Rmetrics — Rmetrics is an Environment and Software Collection for teaching "Financial Engineering and Computational Finance". By Diethelm Wuertz and many others, see the SOURCE file.

fExtremes fExtremes package from Rmetrics. By Diethelm Wuertz and many others, see the SOURCE file.

fOptions fOptions package from Rmetrics. By Diethelm Wuertz and many others, see the SOURCE file.

fSeries fOptions package from Rmetrics. By Diethelm Wuertz and many others, see the SOURCE file.

fortunes R Fortunes. By Achim Zeileis, fortune contributions from Torsten Hothorn, Peter Dalgaard, Uwe Ligges, Kevin Wright.

gap This is an integrated package for genetic data analysis of both population and family data. Currently it contains functions for sample size calculations of both population-based and family-based designs, probability of familial disease aggregation, kinship calculation, some statistics in linkage analysis, and association analysis involving one or more genetic markers including haplotype analysis. In the future it will incorporate programs for path and segregation analyses, as well as other statistics in linkage and association analyses. By Jing hua Zhao in collaboration with other colleagues, and with help from Kurt Hornik and Brian Ripley of the R core development team.

gclus Orders panels in scatterplot matrices and parallel coordinate displays by some merit index. Package contains various indices of merit, ordering functions, and enhanced versions of pairs and parcoord which color panels according to their merit level. By Catherine Hurley.

gpls Classification using generalized partial least squares for two-group and multi-group (more than 2 group) classification. By Beiying Ding.

hapassoc The following R functions are used for likelihood inference of trait associations with haplotypes and other covariates in generalized linear models. The functions accommodate uncertain haplotype phase and can handle missing genotypes at some SNPs. By K. Burkett, B. McNeney, J. Graham.

haplo.stats Haplo Stats is a suite of S-PLUS/R routines for the analysis of indirectly measured haplotypes. The statistical methods assume that all subjects are unrelated and that haplotypes are ambiguous (due to unknown linkage phase of the genetic markers). The genetic markers are assumed to be codominant (i.e., one-to-one correspondence between their genotypes and their phenotypes), and so we refer to the measurements of genetic markers as genotypes. The main functions in Haplo Stats are: `haplo.em`, `haplo.glm` and `haplo.score`. By Jason P. Sinnwell and Daniel J. Schaid.

hett Functions for the fitting and summarizing of heteroscedastic *t*-regression. By Julian Taylor.

httpRequest HTTP Request protocols. Implements the GET, POST and multipart POST request. By Eryk Witold Wolski.

impute Imputation for microarray data (currently KNN only). By Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, Gilbert Chu.

kernlab Kernel-based machine learning methods including support vector machines. By Alexandros Karatzoglou, Alex Smola, Achim Zeileis, Kurt Hornik.

klaR Miscellaneous functions for classification and visualization developed at the Department of Statistics, University of Dortmund. By Christian Roever, Nils Raabe, Karsten Luebke, Uwe Ligges.

knncat This program scales categorical variables in such a way as to make NN classification as accurate as possible. It also handles continuous variables and prior probabilities, and does intelligent variable selection and estimation of error rates and the right number of NN's. By Sam Buttrey.

kza Kolmogorov-Zurbenko Adpative filter for locating change points in a time series. By Brian Close with contributions from Igor Zurbenko.

mAr Estimation of multivariate AR models through a computationally-efficient stepwise least-squares algorithm (Neumaier and Schneider, 2001); the procedure is of particular interest for high-dimensional data without missing values such as geophysical fields. By S. M. Barbosa.

mathgraph Simple tools for constructing and manipulating objects of class mathgraph from the book "S Poetry", available at `http://www.burns-stat.com/pages/spoetry.html`. By Original S code by Patrick J. Burns. Ported to R by Nick Efthymiou.

mcgibbsit Provides an implementation of Warnes & Raftery's MCGibbsit run-length diagnostic for a set of (not-necessarily independent) MCMC sampers. It combines the estimate error-bounding approach of Raftery and Lewis with evaluate between verses within chain approach of Gelman and Rubin. By Gregory R. Warnes.

mixreg Fits mixtures of one-variable regressions (which has been described as doing ANCOVA when you don't know the levels). By Rolf Turner.

mscalib Calibration and filtering methods for calibration of mass spectrometric peptide mass lists. Includes methods for internal, external calibration of mass lists. Provides methods for filtering chemical noise and peptide contaminants. By Eryk Witold Wolski.

multinomRob overdispersed multinomial regression using robust (LQD and tanh) estimation. By Walter R. Mebane, Jr., Jasjeet Singh Sekhon.

mvbutils Utilities for project organization, editing and backup, sourcing, documentation (formal and informal), package preparation, macro functions, and miscellaneous utilities. Needed by **debug** package. By Mark V. Bravington.

mvpart Multivariate regression trees. Original **rpart** by Terry M Therneau and Beth Atkinson, R port by Brian Ripley. Some routines from **vegan** by Jari Oksanen Extensions and adaptations of **rpart** to **mvpart** by Glenn De'ath.

pgam This work is aimed at extending a class of state space models for Poisson count data, so called Poisson-Gamma models, towards a semiparametric specification. Just like the generalized additive models (GAM), cubic splines are used for covariate smoothing. The semiparametric models are fitted by an iterative process that combines maximization of likelihood and backfitting algorithm. By Washington Junger.

qcc Shewhart quality control charts for continuous, attribute and count data. Cusum and EWMA charts. Operating characteristic curves. Process capability analysis. Pareto chart and cause-and-effect chart. By Luca Scrucca.

race Implementation of some racing methods for the empirical selection of the best. If the R package **rpvm** is installed (and if PVM is available, properly configured, and initialized), the evaluation of the candidates are performed in parallel on different hosts. By Mauro Birattari.

rbugs Functions to prepare files needed for running BUGS in batch-mode, and running BUGS from R. Support for Linux systems with Wine is emphasized. By Jun Yan (with part of the code modified from 'bugs.R', `http://www.stat.columbia.edu/~gelman/bugsR/`, by Andrew Gelman).

ref small package with functions for creating references, reading from and writing ro references and a memory efficient refdata type that transparently encapsulates matrixes and data.frames. By Jens Oehlschlägel.

regress Functions to fit Gaussian linear model by maximizing the residual log likelihood. The covariance structure can be written as a linear combination of known matrices. Can be used for multivariate models and random effects models. Easy straight forward manner to specify random effects models, including random interactions. By David Clifford, Peter McCullagh.

rgl 3D visualization device (OpenGL). By Daniel Adler.

rlecuyer Provides an interface to the C implementation of the random number generator with multiple independent streams developed by L'Ecuyer et al (2002). The main purpose of this package is to enable the use of this random number generator in parallel R applications. By Hana Sevcikova, Tony Rossini.

rpart.permutation Performs permutation tests of **rpart** models. By Daniel S. Myers.

rrcov Functions for Robust Location and Scatter Estimation and Robust Regression with High Breakdown Point (`covMcd()`, `ltsReg()`). Originally written for S-PLUS (`fastlts` and `fastmcd`) by Peter Rousseeuw & Katrien van Driessen, ported to R, adapted and packaged by Valentin Todorov.

sandwich Model-robust standard error estimators for time series and longitudinal data. By Thomas Lumley, Achim Zeileis.

seqmon A program that computes the probability of crossing sequential boundaries in a clinical trial. It implements the Armitage-McPherson and Rowe Algorithm using the method described in Schoenfeld D. (2001) " A simple Algorithm for Designing Group Sequential Clinical Trials", *Biometrics* **27**, 972–974. By David A. Schoenfeld.

setRNG Set reproducible random number generator in R and S. By Paul Gilbert.

sfsmisc Useful utilities ['goodies'] from Seminar fuer Statistik ETH Zurich, many ported from S-plus times. By Martin Maechler and many others.

skewt Density, distribution function, quantile function and random generation for the skewed t distribution of Fernandez and Steel. By Robert King,, with contributions from Emily Anderson.

spatialCovariance Functions that compute the spatial covariance matrix for the matern and power classes of spatial models, for data that arise on rectangular units. This code can also be used for the change of support problem and for spatial data that arise on irregularly shaped regions like counties or zipcodes by laying a fine grid of rectangles and aggregating the integrals in a form of Riemann integration. By David Clifford.

spc Evaluation of control charts by means of the zero-state, steady-state ARL (Average Run Length). Setting up control charts for given in-control ARL and plotting of the related figures. The control charts under consideration are one- and two-sided EWMA and CUSUM charts for monitoring the mean of normally distributed independent data. Other charts and parameters are in preparation. Further SPC areas will be covered as well (sampling plans, capability indices, ...). By Sven Knoth.

spe Implements stochastic proximity embedding as described by Agrafiotis et al. in PNAS, 2002, 99, pg 15869 and J. Comput. Chem., 2003,24, pg 1215. By Rajarshi Guha.

supclust Methodology for Supervised Grouping of Predictor Variables. By Marcel Dettling and Martin Maechler.

svmpath Computes the entire regularization path for the two-class SVM classifier with essentialy the same cost as a single SVM fit. By Trevor Hastie.

treeglia Stem analysis functions for volume increment and carbon uptake assessment from tree-rings. By Marco Bascietto.

urca Unit root and cointegration tests encountered in applied econometric analysis are implemented. By Bernhard Pfaff.

urn Functions for sampling without replacement. (Simulated Urns). By Micah Altman.

verify This small package contains simple tools for constructing and manipulating objects of class verify. These are used when regression-testing R software. By S Original by Patrick J. Burns. Ported to R by Nick Efthymiou.

zoo A class with methods for totally ordered indexed observations such as irregular time series. By Achim Zeileis, Gabor Grothendieck.

Other changes

- Packages **mclust1998**, **netCDF**, and **serialize** were moved from the main CRAN section to the Archive.

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
`Kurt.Hornik@R-project.org`

R Foundation News

by Bettina Grün

Donations

- Dipartimento di Scienze Statistiche e Matematiche di Palermo (Italy)
- Emanuele De Rinaldis (Italy)
- Norm Phillips (Canada)

New supporting institutions

- Breast Center at Baylor College of Medicine, Houston, Texas, USA
- Dana-Farber Cancer Institute, Boston, USA
- Department of Biostatistics, Vanderbilt University School of Medicine, USA
- Department of Statistics & Actuarial Science, University of Iowa, USA
- Division of Biostatistics, University of California, Berkeley, USA
- Norwegian Institute of Marine Research, Bergen, Norway
- TERRA Lab, University of Regina - Department of Geography, Canada
- ViaLactia Biosciences (NZ) Ltd, Auckland, New Zealand

New supporting members

Andrej Blejec (Slovenia)
Javier de las Rivas (Spain)
Sandrine Dudoit (USA)
Werner Engl (Austria)
Balazs Györffy (Germany)
Branimir K. Hackenberger (Croatia)
O. Maurice Haynes (USA)
Kai Hendry (UK)
Arne Henningsen (Germany)
Wolfgang Huber (Germany)
Marjatta Kari (Finland)
Osmo Kari (Finland)
Peter Lauf (Germany)
Andy Liaw (USA)
Mu-Yen Lin (Taiwan)
Jeffrey Lins (Danmark)
James W. MacDonald (USA)
Marlene Müller (Germany)
Gillian Raab (UK)
Roland Rau (Germany)
Michael G. Schimek (Austria)
R. Woodrow Setzer (USA)
Thomas Telkamp (Netherlands)
Klaus Thul (Taiwan)
Shusaku Tsumoto (Japan)
Kevin Wright (USA)

Bettina Grün
Technische Universität Wien, Austria
Bettina.Gruen@ci.tuwien.ac.at

Editor-in-Chief:
Thomas Lumley
Department of Biostatistics
University of Washington
Seattle, WA 98195-7232
USA

Editorial Board:
Douglas Bates and Paul Murrell.

Editor Programmer's Niche:
Bill Venables

Editor Help Desk:
Uwe Ligges

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

R News is a publication of the R Foundation for Statistical Computing, communications regarding this publication should be addressed to the editors. All articles are copyrighted by the respective authors. Please send submissions to regular columns to the respective column editor, all other submissions to the editor-in-chief or another member of the editorial board (more detailed submission instructions can be found on the R homepage).

R Project Homepage:
http://www.R-project.org/

This newsletter is available online at
http://CRAN.R-project.org/doc/Rnews/