SOME WORDS ON THE R PROJECT

by Paulo J. Ribeiro Jr. and Patrick. E. Brown
paulojus@est.ufpr.br
p.brown@lancaster.ac.uk

➤ Introduction

The choice of statistical computing environment is a recurring issue and is often a source of conflicting views. The decision of which programme(s) to adopt involves discussion of the needs of teaching, consultancy and/or research, and whether the software is for private use or for adoption at the workplace. The process can involve the intermingling of philosophies of software choice with constraints imposed by availability and finances. There is some consensus that availability of mathematical and statistical tools should be combined with an environment for implementation of new methods. Writing from the econometrics perspective Cribari-Neto and Zarkos (1999) point out that "computer work was mainly a question of reading the manuals and identifying which of the pre-packaged routines would perform the desired task. Times have changed, however, and many newly developed techniques are not available in econometrics packages". Their comments certainly apply to other areas which demand specific techniques or adaptations of the existing ones. From the statistician's perspective there will always be the need for a flexible environment to implement new methods and make them available to a wider audience.

In this article we describe the `R Project` and explain the main reasons why we adopted `R` as our preferred statistical computing environment. These are comments from the user's perspective, and we are not `R` specialists with in-depth knowledge about the inner workings of the system. Patrick has been using `R` since December 1999, judging by the date of his oldest `.RData` file, for both interactive use and computationally intensive batch jobs for his duties as a research associate. Paulo has been using `R` since January 2000 developing a package for geostatistical analysis as part of his PhD work and has run short courses with practical sessions using the package. The text is therefore biased towards `R`'s suitability for our specific research activities in an academic environment. It also reflects our status as computer-literate career statisticians who use the package on a daily basis.

The question we first asked ourselves when writing this article was: *why do we use R ?* The short answer was: it's a good quality statistics package that includes a slew of traditional and modern statistical methods. It's available for a number of operating systems, it's easy to create functions and packages for teaching and research, and the graphics are superb. It interfaces well to lower-level languages such as C, C++ and FORTRAN, and ... it's open source and free!

➤ Platforms

One of the original motivations for the development of `R` was to provide multi-purpose statistical software for the Linux operating system, and `R` was probably the first package to achieve this. Linux, because it is free and open source, has gained popularity not only amongst Unix users, but as an alternative to Windows. As Linux has evolved it has become easier to install and use, and there are currently a number of user-friendly interfaces that come bundled with most Linux distributions. As a consequence the numbers of universities, companies, and private users adopting Linux is increasing.

`R` is well suited to Linux. Many Linux distributions (Mandrake 7.2 for instance) come bundled with `R` (and Matlab's clone: Octave), as well as with many tools that `R` can exploit. `R` is also available for many Windows releases, Macintosh, and Solaris. Since the source code is distributed, `R` can be compiled to run with any platform. (Patrick is waiting for a PalmOS version!)

`R` is accessed solely via a command-line interface, even with the Windows version. This is because, as a non-commercial project, developers had other priorities. Besides, the debate about whether menu-driven interfaces cause more harm than good is still raging. Seasoned programmers will appreciate `R`'s terse interface, as it lends itself to speed and flexibility, but users of Excel and others menu driven software may find the

solitary "`>`" prompt intimidating. There is an attempt to develop a GNOME interface but this is still in its early stages. For Emacs users, the functionality of ESS (*Emacs Speaks Statistics*) includes support for `R`.

➤ So, what is R?

Quoting the `R` documentation: "`R` is a computer language not entirely unlike the `S` language developed at AT&T Bell Laboratories by Rick Becker, John Chambers and Allan Wilks. The two languages are implemented quite differently, but bear enough superficial resemblance that users should be able to switch between the two with relative ease." `R` can be regarded as a re-implementation of the `S` language. Motivations for its development include an attempt at a more efficient implementation of the `S` language, specially concerning memory demands, and development for platforms not supported by `S-PLUS` (at least at that time) like Linux and Macintoch. `R` was first announced on the *S-News* in August, 1993. Published reference about the project goes back to Ihaka and Gentleman (1996) reporting the initial development at the Statistics Department of the University of Auckland. Since 1995 `R` has been distributed as a free software with beta versions available and constantly updated. The first non-beta version (`R-1.0`) was released on 29th February, 2000. The version 1.2 introduced significant changes in the memory management. The main consequence is the workspace is no longer static, it can grow and shrink as needed, freeing the user of the obligation to anticipate and allocate necessary memory for each session. The current version is `R-1.2.1` available at CRAN (Comprehensive R Archive Network).

`R` consists of two basic parts: *R-base*, with the main code, and the *contributed packages*, a collection of codes implementing a diversity of statistical methods. In `R`'s terminology the word *package* is used instead of `S-PLUS's` *library*. `R` development and policy are coordinated by the `R` *core team*, currently consisting of Douglas Bates, John Chambers, Peter Dalgaard, Robert Gentleman, Kurt Hornik, Ross Ihaka, Friedrich Leisch, Thomas Lumley, Martin Maechler, Guido Masarotto, Paul Murrell, Brian Ripley, Duncan Temple Lang, Luke Tierney. The core team was established in mid-1997 and plays the central role in `R` development and distribution, including responsibility for changes in the basic code. The project as a whole has contributions from a much larger group of people. The names of the main contributors are listed by typing `contributors()` at the `R` prompt.

For those of you unfamiliar with both, `R` and `S-PLUS` are packages which provide facilities for data manipulation, calculation and graphical display. These include collections of tools for data analysis and a simple programming language which can be used to implement new methods of data analysis and/or to customise existing ones. A key distinction from other statistical software is the emphasis on doing analyses in steps, storing results in objects which can be processed and interrogated by other functions. `R` is a run-time environment, meaning that code is not compiled into executables the way C code, for instance, is. A short example of `R` code is:

```
xyregression <- lm(y ~ x)
qqnorm(xyregression$resid)
printgraph(file="qqnorm.ps")
```

The first command performs a linear regression on the data in vectors x and y, saving the results as an object called "xyregression"; the second plots the normal scores of the residuals in a graphics window and the third saves the graph as a postscript file.

➤ `R` and `S-PLUS`: are they different?

The most obvious difference between `R` and `S-PLUS` is that `R` is an open-source project which distributes its source code under the terms of the GNU public license. The first advantage of open-source software is that it is free, freeing up a department's software license budget to fund more conferences overseas for postgrads and researchers! In fact, free software might be the only viable alternative for several departments around the world which cannot afford annual licenses for good quality statistical software.

On a scientific note, open source software allows the user to inspect and study the source code, so we'll always know exactly which eigenvalue algorithm is used to compute our principle component analyses. Furthermore, the

source code can be modified and re-distributed, or compiled as a stand-alone executable, provided the new code is also made publicly available.

The primary differences between the two packages are not on the surface, but on how the programmes function internally. Although to a user the programmes are almost identical, the bones of the packages are quite different. The root of the difference is that R uses what clever people call *lexical scoping*. This feature is inherited from the *Scheme* programming language from which R developers have borrowed many ideas. S-PLUS only uses local and global variables, whereas R has a hierarchy of variables whereby a function defined within a function can see variables defined in the first function. One result of the differing implementations is that R manages its own memory, allowing loops to operate more quickly than S-PLUS, reducing the critical problem of exploding memory usage. Ripley (2001) point out that: "R is more tolerant of badly-written code which can make S-PLUS slow to crawl". We both have had programmes that use too much memory to be run in S-PLUS but run without problems in R ...

For the average user the two packages appear almost identical. Changing the `motif()` command to `X11()` is sufficient to run most simple S-PLUS code in R. One important difference is that while S-PLUS saves each object as a separate file, R saves the entire workspace as one file, the way Matlab does. Transferring data between R and S-PLUS is usually straightforward with the `dump` and `restore` commands. There is no analogue of S-PLUS's Trellis on R. On the other hand R's plots accept mathematical notation and have more versatile colors schemes, at least when using the command line under Linux. Yet another difference is the extra flexibility in algorithms for random number generation: there are more algorithms available in R (and being open source they can be re-implemented in other engines). Other differences are listed in R's FAQ and Venables and Ripley's (1999) on-line complements. We will leave them to the reader. Compatibility with S has been pursued but sometimes has to be sacrificed in attempts to repair short-comings of S.

We particularly like the tools to create packages available to the R programmer. Documentation can be written using a, LaTeX style, pre-formated type of document. A host of perl scripts assemble the code, check the example functions, build and convert documentation to a number of formats (*pdf* and *HTML*, among others). Scripts are available to convert to and from S-PLUS formats. Quite often we find packages originally written for one of the engines and converted to the other. Building a package like `geoR` say, under Linux, is done by: writing functions and documentation, copying them and other source code to a directory `geoR` with a specific structure, writing a short DESCRIPTION file and typing the shell command

```
R CMD build geoR
```

The resulting file can be distributed and installed by typing:

```
R INSTALL
geoR_version.number.tar.gz
```

For packages available at CRAN the installation and update is even easier. It can be done online during an R session using the functions `install.packages()` and `update.packages()`.

➤ Teaching

R is ideal for teaching for a number of reasons. Since it is accessed by programming rather than menus, students can perform linear regression by typing:

```
betahat <- solve(t(x) %*% x)
%*% t(x) %*% y
```

and can be stimulated to write their own code, as well as to inspect the ones already available.

Being free, R can be distributed and installed on personal computers. Class examples can be reproduced at home regardless of which engine is used at the University Labs. Packages can be easily prepared providing excellent and well organized material for courses. Actually, much of the available resources in R comes from teaching material. A very nice example of teaching introductory courses using R is given by Nolan and Speed (2000). An experience with a web-based interface is described by M.J. Ray in the first issue of the `R News`. The article describes an implementation of a *Rcgi* interface at the University of East Anglia. Analyses can be performed remotely. Input is passed from a web-browser to

the program running on a server and output is passed back again to the web-browser.

➤ Integration

As programming tools R and S are easier to learn and quicker to program though not as efficient as languages like C, C++ or Fortran. However, one of the virtues of both is the possibility of integration with other codes. Shared libraries can be loaded, opening the possibility of using R and S-PLUS for graphical interface even if the number-crunching uses a lower-level languages.

Other kinds of integration and interfaces have been discussed and implemented in R. Some examples are alternatives to distributed computing, interfaces with database management systems and communications with other languages/applications. Probably, these are areas where we will see most of the further developments. We refer to the first issue of the *R-News*

```
cran.r-project.org/doc/Rnews/
```

for more information on specific projects.

➤ Bayesian methods in R

Although the current distribution does not include many Bayesian tools, other available resources, including integration with other languages, make R a suitable environment for implementing Bayesian methods. The package CODA (output analysis and diagnostics for MCMC) by Martyn Plummer, Nicky Best, Kate Cowles and Karen Vines, included as a contributed package in CRAN, is an excellent tool for Bayesians needing to convince referees that their chains have indeed converged. Paulo has implemented some Bayesian methods for spatial data in the geoR package. There are certainly more Bayesian packages available that we will be chastised for not mentioning.

There have been messages on the topic of Bayesian methods on R's help list and it is clear that there is space for further development, either in the form of additional R packages or by integrating R with existing Bayesian software. Hopefully in the future there will be more resources at CRAN implementing Bayesian methods.

➤ Documentation and Support

The official R's web sites are:

```
www.r-project.org
cran.r-project.org
```

The first is "the" R home page while the second works as a download area with several mirrors. Extensive and detailed documentation can be found at the sites, and are also included in the software distributions. Some examples of available documents are ``An Introduction to R'' and ``Writing R Extensions''. Specific documentation for the packages can be found at the contributed packages web-page. Many basic questions are answered in R's FAQ. Functions are documented and help on specific functions are obtained by typing

```
help(function.name).
```

The function help.start() allows visualisation in HTML format. The only support available is through the *R help list* and we can only report positively from our experience subscribing to it.

➤ How to start?

The best way to start is to be motivated by an ongoing project. It's worth looking at the *demos* available by typing at the command line

```
demo()
```

and then selecting one of the available options. Try demo(graphics), for example. Users not familiar with S-PLUS can start running the examples listed in the Appendix A of ``An Introduction to R''. This will give a good idea of the R environment and the basic resources available, before start reading more detailed material.

Given the similarities with S-PLUS, most of the literature can be shared. For example the "classic" books by Becker, Chambers and Wilks (1998) and Chambers and Hastie (1988) can be adapted for R. Venables and Ripley (1999) has online complements on R and Venables and Ripley (2000) is written such that R's specific features are highlighted.

➤ References

BECKER, R.A., CHAMBERS, J.M. and WILKS, A.R. (1988) *The New S Language*. Chapman & Hall.

CHAMBERS, J.M. and HASTIE, T.J. (1992) *Statistical Models in S*. Chapman & Hall.

CRIBARI-NETO, F. and ZARKOS, S.G. (1999) R: yet another econometric programming environment. *Journal of Applied Econometrics*, **14**, 319–329.

IHAKA, R. and GENTLEMAN, R. (1996) R: a language for data

analysis and graphics. *Journal of Computational and Graphical Statistics*, **5**, 299–315.

NOLAN, D. and SPEED, T.D. (2000) *Stats Labs: Mathematical Statistics Through Application*. Springer.

RIPLEY, B.D. (2001) The R project in statistical computing. *LTSN Math, Stats & OR Network Newsletter* No 5 (March).

VENABLES, W.N. and RIPLEY, B.D. (1999) *Modern Applied Statistics with S-PLUS*. 3rd ed. Springer.

VENABLES, W.N. and RIPLEY, B.D. (2000) *S Programming*. Springer.