R FAQ

Frequently Asked Questions on R
Version 1.8-17, 2003-11-15
ISBN 3-900051-01-1

Kurt Hornik

Table of Contents

1 Introduction

This document contains answers to some of the most frequently asked questions about R.

1.1 Legalese

1.2 Obtaining this document

The latest version of this document is always available from

http://www.ci.tuwien.ac.at/~hornik/R/

From there, you can obtain versions converted to plain ASCII text, DVI, GNU info, HTML, PDF, PostScript as well as the Texinfo source used for creating all these formats using the GNU Texinfo system.

You can also obtain the R FAQ from the '`doc/FAQ`' subdirectory of a CRAN site (see Section 2.10 [What is CRAN?], page 9).

1.3 Citing this document

In publications, please refer to this FAQ as Hornik (2003), "The R FAQ", and give the above, *official* URL and the ISBN 3-900051-01-1.

1.4 Notation

Everything should be pretty standard. '`R>`' is used for the R prompt, and a '`$`' for the shell prompt (where applicable).

1.5 Feedback

Feedback is of course most welcome.

In particular, note that I do not have access to Windows or Macintosh systems. Features specific to the Windows and MacOS X ports of R are described in the "R for Windows FAQ" and the "R for Macintosh FAQ/DOC". If you have information on Macintosh or Windows systems that you think should be added to this document, please let me know.

2 R Basics

2.1 What is R?

R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.

The design of R has been heavily influenced by two existing languages: Becker, Chambers & Wilks' S (see Section 3.1 [What is S?], page 11) and Sussman's Scheme. Whereas the resulting language is very similar in appearance to S, the underlying implementation and semantics are derived from Scheme. See Section 3.3 [What are the differences between R and S?], page 12, for further details.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. Most of the user-visible functions in R are written in R. It is possible for the user to interface to procedures written in the C, C++, or FORTRAN languages for efficiency. The R distribution contains functionality for a large number of statistical procedures. Among these are: linear and generalized linear models, nonlinear regression models, time series analysis, classical parametric and nonparametric tests, clustering and smoothing. There is also a large set of functions which provide a flexible graphical environment for creating various kinds of data presentations. Additional modules ("add-on packages") are available for a variety of specific purposes (see Chapter 5 [R Add-On Packages], page 20).

R was initially written by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand. In addition, a large group of individuals has contributed to R by sending code and bug reports.

Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code CVS archive. The group currently consists of Doug Bates, John Chambers, Peter Dalgaard, Robert Gentleman, Kurt Hornik, Stefano Iacus, Ross Ihaka, Friedrich Leisch, Thomas Lumley, Martin Maechler, Duncan Murdoch, Paul Murrell, Martyn Plummer, Brian Ripley, Duncan Temple Lang, and Luke Tierney.

R has a home page at `http://www.R-project.org/`. It is free software distributed under a GNU-style copyleft, and an official part of the GNU project ("GNU S").

2.2 What machines does R run on?

R is being developed for the Unix, Windows and Mac families of operating systems. Support for Mac OS Classic will end with the 1.7 series.

The current version of R will configure and build under a number of common Unix platforms including i386-freebsd, *cpu*-linux-gnu for the i386, alpha, arm, hppa, ia64, m68k, powerpc, and sparc CPUs (see e.g. `http://buildd.debian.org/build.php?&pkg=r-base`), i386-sun-solaris, powerpc-apple-darwin, mips-sgi-irix, alpha-dec-osf4, rs6000-ibm-aix, hppa-hp-hpux, and sparc-sun-solaris.

If you know about other platforms, please drop us a note.

2.3 What is the current version of R?

The current released version is 1.8.0. Based on this 'major.minor.patchlevel' numbering scheme, there are two development versions of R, working towards the next patch ('r-patched') and minor or eventually major ('r-devel') releases of R, respectively. Version r-patched is for bug fixes mostly. New features are typically introduced in r-devel.

2.4 How can R be obtained?

Sources, binaries and documentation for R can be obtained via CRAN, the "Comprehensive R Archive Network" (see Section 2.10 [What is CRAN?], page 9).

Sources are also available via anonymous rsync. Use

```
rsync -rC --delete rsync.R-project.org::module R
```

to create a copy of the source tree specified by *module* in the subdirectory '`R`' of the current directory, where *module* specifies one of the three existing flavors of the R sources, and can be one of '`r-release`' (current released version), '`r-patched`' (patched released version), and '`r-devel`' (development version). The rsync trees are created directly from the master CVS archive and are updated hourly. The '`-C`' and in the `rsync` command is to cause it to skip the CVS directories. Further information on `rsync` is available at http://rsync.samba.org/rsync/.

The sources of the development version are also available via anonymous CVS. See http://anoncvs.R-project.org for more information.

2.5 How can R be installed?

2.5.1 How can R be installed (Unix)

If binaries are available for your platform (see Section 2.6 [Are there Unix binaries for R?], page 6), you can use these, following the instructions that come with them.

Otherwise, you can compile and install R yourself, which can be done very easily under a number of common Unix platforms (see Section 2.2 [What machines does R run on?], page 3). The file '`INSTALL`' that comes with the R distribution contains a brief introduction, and the "R Installation and Administration" guide (see Section 2.7 [What documentation exists for R?], page 6) has full details.

Note that you need a FORTRAN compiler or `f2c` in addition to a C compiler to build R. Also, you need Perl version 5 to build the R object documentations. (If this is not available on your system, you can obtain a PDF version of the object reference manual via CRAN.)

In the simplest case, untar the R source code, change to the directory thus created, and issue the following commands (at the shell prompt):

```
$ ./configure
$ make
```

If these commands execute successfully, the R binary and a shell script front-end called '`R`' are created and copied to the '`bin`' directory. You can copy the script to a place where

users can invoke it, for example to '`/usr/local/bin`'. In addition, plain text help pages as well as HTML and LaTeX versions of the documentation are built.

Use *`make dvi`* to create DVI versions of the R manuals, such as '`refman.dvi`' (an R object reference index) and '`R-exts.dvi`', the "R Extension Writers Guide", in the '`doc/manual`' subdirectory. These files can be previewed and printed using standard programs such as `xdvi` and `dvips`. You can also use *`make pdf`* to build PDF (Portable Document Format) version of the manuals, and view these using e.g. Acrobat. Manuals written in the GNU Texinfo system can also be converted to info files suitable for reading online with Emacs or stand-alone GNU Info; use *`make info`* to create these versions (note that this requires `makeinfo` version 4).

Finally, use *`make check`* to find out whether your R system works correctly.

You can also perform a "system-wide" installation using *`make install`*. By default, this will install to the following directories:

'`${prefix}/bin`'
: the front-end shell script

'`${prefix}/man/man1`'
: the man page

'`${prefix}/lib/R`'
: all the rest (libraries, on-line help system, ...). This is the "R Home Directory" (`R_HOME`) of the installed system.

In the above, `prefix` is determined during configuration (typically '`/usr/local`') and can be set by running `configure` with the option

```
$ ./configure --prefix=/where/you/want/R/to/go
```

(E.g., the R executable will then be installed into '`/where/you/want/R/to/go/bin`'.)

To install DVI, info and PDF versions of the manuals, use *`make install-dvi`*, *`make install-info`* and *`make install-pdf`*, respectively.

2.5.2 How can R be installed (Windows)

The '`bin/windows`' directory of a CRAN site contains binaries for a base distribution and a large number of add-on packages from CRAN to run on Windows 95, 98, ME, NT4, 2000, and XP (at least) on Intel and clones (but not on other platforms). The Windows version of R was created by Robert Gentleman, and is now being developed and maintained by Duncan Murdoch and Brian D. Ripley.

For most installations the Windows installer program will be the easiest tool to use.

See the "R for Windows FAQ" for more details.

2.5.3 How can R be installed (Macintosh)

The '`bin/macosx`' directory of a CRAN site contains a standard Apple installer package named '`RAqua.pkg.sit`' compressed in Aladdin Stuffit format. Once downloaded, uncompressed and executed, the installer will install the current non-developer release of R. RAqua is a native MacOSX Darwin version of R with an Aqua GUI. Inside '`bin/macosx/x.y`' there

are prebuilt binary packages to be used with RAqua corresponding to the "*x.y*" release of R. The installation of these packages is available through the "Package" menu of the RAqua GUI. This port of R for MacOSX is maintained by Stefano Iacus. The "R for Macintosh FAQ/DOC" has more details.

The '`bin/macos`' directory of a CRAN site contains bin-hexed ('`hqx`') and stuffit ('`sit`') archives for a base distribution and a large number of add-on packages of R 1.7.1 to run under MacOS 8.6 to MacOS 9.2.2. This port of R for Macintosh is no longer supported.

2.6 Are there Unix binaries for R?

The '`bin/linux`' directory of a CRAN site contains Debian stable/testing packages for the i386 platform (now part of the Debian distribution and maintained by Dirk Eddelbuettel), Mandrake 9.0/9.1 i386 packages by Michele Alzetta, Red Hat 7.x/8.x/9 i386 packages by Martyn Plummer, SuSE 7.3/8.0/8.1/8.2/9.0 i386 packages by Detlef Steuer, and VineLinux 2.6 i386 packages by Susunu Tanimura.

The Debian packages can be accessed through APT, the Debian package maintenance tool. Simply add the line

```
deb http://cran.R-project.org/bin/linux/debian distribution main
```

(where *distribution* is either '`stable`' or '`testing`'; feel free to use a CRAN mirror instead of the master) to the file '`/etc/apt/sources.list`'. Once you have added that line the programs `apt-get`, `apt-cache`, and `dselect` (using the apt access method) will automatically detect and install updates of the R packages.

No other binary distributions are currently publically available.

2.7 What documentation exists for R?

Online documentation for most of the functions and variables in R exists, and can be printed on-screen by typing `help(name)` (or `?name`) at the R prompt, where *name* is the name of the topic help is sought for. (In the case of unary and binary operators and control-flow special forms, the name may need to be be quoted.)

This documentation can also be made available as one reference manual for on-line reading in HTML and PDF formats, and as hardcopy via LaTeX, see Section 2.5 [How can R be installed?], page 4. An up-to-date HTML version is always available for web browsing at `http://stat.ethz.ch/R-manual/`.

The R distribution also comes with the following manuals.

- "An Introduction to R" ('`R-intro`') includes information on data types, programming elements, statistical modeling and graphics. This document is based on the "Notes on S-Plus" by Bill Venables and David Smith.
- "Writing R Extensions" ('`R-exts`') currently describes the process of creating R add-on packages, writing R documentation, R's system and foreign language interfaces, and the R API.
- "R Data Import/Export" ('`R-data`') is a guide to importing and exporting data to and from R.

- "The R Language Definition" ('`R-lang`'), a first version of the "Kernighan & Ritchie of R", explains evaluation, parsing, object oriented programming, computing on the language, and so forth.
- "R Installation and Administration" ('`R-admin`').

Books on R include

P. Dalgaard (2002), "Introductory Statistics with R", Springer: New York, ISBN 0-387-95475-9.

J. Fox (2002), "An R and S-PLUS Companion to Applied Regression", Sage Publications, ISBN 0-761-92280-6 (softcover) or 0-761-92279-2 (hardcover), http://socserv.socsci.mcmaster.ca/jfox/Books/Companion/.

J. Maindonald and J. Braun (2003), "Data Analysis and Graphics Using R: An Example-Based Approach", Cambridge University Press, ISBN 0-521-81336-0, http://wwwmaths.anu.edu.au/~johnm/.

S. M. Iacus and G. Masarotto (2002), "Laboratorio di statistica con R ", McGraw-Hill, ISBN 88-386-6084-0 (in Italian).

The book

W. N. Venables and B. D. Ripley (2002), "Modern Applied Statistics with S. Fourth Edition". Springer, ISBN 0-387-95457-0

has a home page at http://www.stats.ox.ac.uk/pub/MASS4/ providing additional material. Its companion is

W. N. Venables and B. D. Ripley (2000), "S Programming". Springer, ISBN 0-387-98966-8

and provides an in-depth guide to writing software in the S language which forms the basis of both the commercial S-PLUS and the Open Source R data analysis software systems. See http://www.stats.ox.ac.uk/pub/MASS3/Sprog/ for more information.

In addition to material written specifically or explicitly for R, documentation for S/S-PLUS (see Chapter 3 [R and S], page 11) can be used in combination with this FAQ (see Section 3.3 [What are the differences between R and S?], page 12). Introductory books include

P. Spector (1994), "An introduction to S and S-PLUS", Duxbury Press.

A. Krause and M. Olsen (2002), "The Basics of S-PLUS" (Third Edition). Springer, ISBN 0-387-95456-2

The book

J. C. Pinheiro and D. M. Bates (2000), "Mixed-Effects Models in S and S-PLUS", Springer, ISBN 0-387-98957-0

provides a comprehensive guide to the use of the **nlme** package for linear and nonlinear mixed-effects models. This has a home page at http://nlme.stat.wisc.edu/MEMSS/.

As an example of how R can be used in teaching an advanced introductory statistics course, see

D. Nolan and T. Speed (2000), "Stat Labs: Mathematical Statistics Through Applications", Springer Texts in Statistics, ISBN 0-387-98974-9

This integrates theory of statistics with the practice of statistics through a collection of case studies ("labs"), and uses R to analyze the data. More information can be found at http://www.stat.Berkeley.EDU/users/statlabs/.

Last, but not least, Ross' and Robert's experience in designing and implementing R is described in Ihaka & Gentleman (1996), "R: A Language for Data Analysis and Graphics", *Journal of Computational and Graphical Statistics*, **5**, 299–314.

An annotated bibliography (BibTEX format) of R-related publications which includes most of the above references can be found at

http://www.R-project.org/doc/bib/R.bib

2.8 Citing R

To cite R in publications, use

```
@Manual{,
  title        = {R: A language and environment for statistical
                  computing},
  author       = {{R Development Core Team}},
  organization = {R Foundation for Statistical Computing},
  address      = {Vienna, Austria},
  year         = 2003,
  note         = {ISBN 3-900051-00-3},
  url          = {http://www.R-project.org}
}
```

2.9 What mailing lists exist for R?

Thanks to Martin Maechler, there are four mailing lists devoted to R.

`R-announce`
: A moderated list for announcements about the development of R and the availability of new code.

`R-packages`
: A moderated list for announcements on the availability of new or enhanced contributed packages.

`R-help`
: The 'main' R mailing list, for discussion about problems and solutions using R, announcements (not covered by 'R-announce' and 'R-packages') about the development of R and the availability of new code, enhancements and patches to the source code and documentation of R, comparison and compatibility with S and S-PLUS, and for the posting of nice examples and benchmarks.

`R-devel`
: This list is for discussions about the future of R and pre-testing of new versions. It is meant for those who maintain an active position in the development of R.

Note that the R-announce and R-packages lists are gatewayed into R-help. Hence, you should subscribe to either of them only in case you are not subscribed to R-help.

Send email to R-help@lists.R-project.org to reach everyone on the R-help mailing list. To subscribe (or unsubscribe) to this list send '`subscribe`' (or '`unsubscribe`') in the *body* of the message (not in the subject!) to R-help-request@lists.R-project.org. Information about the list can be obtained by sending an email with '`info`' as its contents to R-help-request@lists.R-project.org.

Subscription and posting to the other lists is done analogously, with '`R-help`' replaced by '`R-announce`', '`R-packages`', and '`R-devel`', respectively.

Subscriptions to the R-help and R-devel mailing lists are also available in digest (plain or MIME) format, see the '`doc/html/mail.html`' file in CRAN for more information.

It is recommended that you send mail to R-help rather than only to the R Core developers (who are also subscribed to the list, of course). This may save them precious time they can use for constantly improving R, and will typically also result in much quicker feedback for yourself.

Of course, in the case of bug reports it would be very helpful to have code which reliably reproduces the problem. Also, make sure that you include information on the system and version of R being used. See Chapter 9 [R Bugs], page 55 for more details.

Archives of the above three mailing lists are made available on the net in a monthly schedule via the '`doc/html/mail.html`' file in CRAN. Searchable archives of the lists are available via http://maths.newcastle.edu.au/~rking/R/.

The R Core Team can be reached at R-core@lists.R-project.org for comments and reports.

2.10 What is CRAN?

The "Comprehensive R Archive Network" (CRAN) is a collection of sites which carry identical material, consisting of the R distribution(s), the contributed extensions, documentation for R, and binaries.

The CRAN master site at TU Wien, Austria, can be found at the URL

http://cran.R-project.org/

and is currently being mirrored daily at

http://cran.at.R-project.org/	(TU Wien, Austria)
http://cran.au.R-project.org/	(PlanetMirror, Australia)
http://cran.br.R-project.org/	(Universidade Federal de Paraná, Brazil)
http://cran.ch.R-project.org/	(ETH Zürich, Switzerland)
http://cran.de.R-project.org/	(APP, Germany)
http://cran.dk.R-project.org/	(SunSITE, Denmark)
http://cran.es.R-project.org/	(Spanish National Research Network, Madrid, Spain)
http://cran.hu.R-project.org/	(Semmelweis U, Hungary)
http://cran.uk.R-project.org/	(U of Bristol, United Kingdom)
http://cran.us.R-project.org/	(U of Wisconsin, USA)
http://cran.za.R-project.org/	(Rhodes U, South Africa)

Please use the CRAN site closest to you to reduce network load.

From CRAN, you can obtain the latest official release of R, daily snapshots of R (copies of the current CVS trees), as gzipped and bzipped tar files, a wealth of additional contributed code, as well as prebuilt binaries for various operating systems (Linux, MacOS Classic, MacOS X, and MS Windows). CRAN also provides access to documentation on R, existing mailing lists and the R Bug Tracking system.

To "submit" to CRAN, simply upload to `ftp://cran.R-project.org/incoming/` and send an email to `cran@R-project.org`. Note that CRAN generally does not accept submissions of precompiled binaries due to security reasons.

> **Note:** It is very important that you indicate the copyright (license) information (GPL, BSD, Artistic, ...) in your submission.

Please always use the URL of the master site when referring to CRAN.

2.11 Can I use R for commercial purposes?

R is released under the GNU General Public License (GPL). If you have any questions regarding the legality of using R in any particular situation you should bring it up with your legal counsel. We are in no position to offer legal advice.

It is the opinion of the R Core Team that one can use R for commercial purposes (e.g., in business or in consulting). The GPL, like all Open Source licenses, permits all and any use of the package. It only restricts distribution of R or of other programs containing code from R. This is made clear in clause 6 ("No Discrimination Against Fields of Endeavor") of the Open Source Definition:

> The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

It is also explicitly stated in clause 0 of the GPL, which says in part

> Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program.

Most add-on packages, including all recommended ones, also explicitly allow commercial use in this way. A few packages are restricted to "non-commercial use"; you should contact the author to clarify whether these may be used or seek the advice of your legal counsel.

None of the discussion in this section constitutes legal advice. The R Core Team does not provide legal advice under any circumstances.

2.12 Why is R named R?

The name is partly based on the (first) names of the first two R authors (Robert Gentleman and Ross Ihaka), and partly a play on the name of the Bell Labs language 'S' (see Section 3.1 [What is S?], page 11).

3 R and S

3.1 What is S?

S is a very high level language and an environment for data analysis and graphics. In 1998, the Association for Computing Machinery (ACM) presented its Software System Award to John M. Chambers, the principal designer of S, for

> the S system, which has forever altered the way people analyze, visualize, and manipulate data . . .
>
> S is an elegant, widely accepted, and enduring software system, with conceptual integrity, thanks to the insight, taste, and effort of John Chambers.

The evolution of the S language is characterized by four books by John Chambers and coauthors, which are also the primary references for S.

- Richard A. Becker and John M. Chambers (1984), "S. An Interactive Environment for Data Analysis and Graphics," Monterey: Wadsworth and Brooks/Cole.

  This is also referred to as the "*Brown Book*", and of historical interest only.
- Richard A. Becker, John M. Chambers and Allan R. Wilks (1988), "The New S Language," London: Chapman & Hall.

  This book is often called the "*Blue Book*", and introduced what is now known as S version 2.
- John M. Chambers and Trevor J. Hastie (1992), "Statistical Models in S," London: Chapman & Hall.

  This is also called the "*White Book*", and introduced S version 3, which added structures to facilitate statistical modeling in S.
- John M. Chambers (1998), "Programming with Data," New York: Springer, ISBN 0-387-98503-4 (`http://cm.bell-labs.com/cm/ms/departments/sia/Sbook/`).

  This "*Green Book*" describes version 4 of S, a major revision of S designed by John Chambers to improve its usefulness at every stage of the programming process.

See `http://cm.bell-labs.com/cm/ms/departments/sia/S/history.html` for further information on "Stages in the Evolution of S".

There is a huge amount of user-contributed code for S, available at the S Repository at CMU.

3.2 What is S-PLUS?

S-PLUS is a value-added version of S sold by Insightful Corporation. Based on the S language, S-PLUS provides functionality in a wide variety of areas, including robust regression, modern non-parametric regression, time series, survival analysis, multivariate analysis, classical statistical tests, quality control, and graphics drivers. Add-on modules add additional capabilities for wavelet analysis, spatial statistics, GARCH models, and design of experiments.

See the Insightful S-PLUS page for further information.

3.3 What are the differences between R and S?

We can regard S as a language with three current implementations or "engines", the "old S engine" (S version 3; S-PLUS 3.x and 4.x), the "new S engine" (S version 4; S-PLUS 5.x and above), and R. Given this understanding, asking for "the differences between R and S" really amounts to asking for the specifics of the R implementation of the S language, i.e., the difference between the R and S *engines*.

For the remainder of this section, "S" refers to the S engines and not the S language.

3.3.1 Lexical scoping

Contrary to other implementations of the S language, R has adopted the evaluation model of Scheme.

This difference becomes manifest when *free* variables occur in a function. Free variables are those which are neither formal parameters (occurring in the argument list of the function) nor local variables (created by assigning to them in the body of the function). Whereas S (like C) by default uses *static* scoping, R (like Scheme) has adopted *lexical* scoping. This means the values of free variables are determined by a set of global variables in S, but in R by the bindings that were in effect at the time the function was created.

Consider the following function:

```
cube <- function(n) {
  sq <- function() n * n
  n * sq()
}
```

Under S, `sq()` does not "know" about the variable `n` unless it is defined globally:

```
S> cube(2)
Error in sq():  Object "n" not found
Dumped
S> n <- 3
S> cube(2)
[1] 18
```

In R, the "environment" created when `cube()` was invoked is also looked in:

```
R> cube(2)
[1] 8
```

As a more "interesting" real-world problem, suppose you want to write a function which returns the density function of the r-th order statistic from a sample of size n from a (continuous) distribution. For simplicity, we shall use both the cdf and pdf of the distribution as explicit arguments. (Example compiled from various postings by Luke Tierney.)

The S-PLUS documentation for `call()` basically suggests the following:

```
dorder <- function(n, r, pfun, dfun) {
  f <- function(x) NULL
  con <- round(exp(lgamma(n + 1) - lgamma(r) - lgamma(n - r + 1)))
  PF <- call(substitute(pfun), as.name("x"))
  DF <- call(substitute(dfun), as.name("x"))
  f[[length(f)]] <-
    call("*", con,
         call("*", call("^", PF, r - 1),
              call("*", call("^", call("-", 1, PF), n - r),
                   DF)))
  f
}
```

Rather tricky, isn't it? The code uses the fact that in S, functions are just lists of special mode with the function body as the last argument, and hence does not work in R (one could make the idea work, though).

A version which makes heavy use of `substitute()` and seems to work under both S and R is

```
dorder <- function(n, r, pfun, dfun) {
  con <- round(exp(lgamma(n + 1) - lgamma(r) - lgamma(n - r + 1)))
  eval(substitute(function(x) K * PF(x)^a * (1 - PF(x))^b * DF(x),
                  list(PF = substitute(pfun), DF = substitute(dfun),
                       a = r - 1, b = n - r, K = con)))
}
```

(the `eval()` is not needed in S).

However, in R there is a much easier solution:

```
dorder <- function(n, r, pfun, dfun) {
  con <- round(exp(lgamma(n + 1) - lgamma(r) - lgamma(n - r + 1)))
  function(x) {
    con * pfun(x)^(r - 1) * (1 - pfun(x))^(n - r) * dfun(x)
  }
}
```

This seems to be the "natural" implementation, and it works because the free variables in the returned function can be looked up in the defining environment (this is lexical scope).

Note that what you really need is the function *closure*, i.e., the body along with all variable bindings needed for evaluating it. Since in the above version, the free variables in the value function are not modified, you can actually use it in S as well if you abstract out the closure operation into a function `MC()` (for "make closure"):

```
dorder <- function(n, r, pfun, dfun) {
  con <- round(exp(lgamma(n + 1) - lgamma(r) - lgamma(n - r + 1)))
  MC(function(x) {
       con * pfun(x)^(r - 1) * (1 - pfun(x))^(n - r) * dfun(x)
     },
     list(con = con, pfun = pfun, dfun = dfun, r = r, n = n))
}
```

Given the appropriate definitions of the closure operator, this works in both R and S, and is much "cleaner" than a substitute/eval solution (or one which overrules the default scoping rules by using explicit access to evaluation frames, as is of course possible in both R and S).

For R, `MC()` simply is

```
MC <- function(f, env) f
```

(lexical scope!), a version for S is

```
MC <- function(f, env = NULL) {
  env <- as.list(env)
  if (mode(f) != "function")
    stop(paste("not a function:", f))
  if (length(env) > 0 && any(names(env) == ""))
    stop(paste("not all arguments are named:", env))
  fargs <- if(length(f) > 1) f[1:(length(f) - 1)] else NULL
  fargs <- c(fargs, env)
  if (any(duplicated(names(fargs))))
    stop(paste("duplicated arguments:", paste(names(fargs)),
         collapse = ", "))
  fbody <- f[length(f)]
  cf <- c(fargs, fbody)
  mode(cf) <- "function"
  return(cf)
}
```

Similarly, most optimization (or zero-finding) routines need some arguments to be optimized over and have other parameters that depend on the data but are fixed with respect to optimization. With R scoping rules, this is a trivial problem; simply make up the function with the required definitions in the same environment and scoping takes care of it. With S, one solution is to add an extra parameter to the function and to the optimizer to pass in these extras, which however can only work if the optimizer supports this.

Lexical scoping allows using function closures and maintaining local state. A simple example (taken from Abelson and Sussman) is obtained by typing *`demo("scoping")`* at the R prompt. Further information is provided in the standard R reference "R: A Language for Data Analysis and Graphics" (see Section 2.7 [What documentation exists for R?], page 6) and in Robert Gentleman and Ross Ihaka (2000), "Lexical Scope and Statistical Computing", *Journal of Computational and Graphical Statistics*, **9**, 491–508.

Lexical scoping also implies a further major difference. Whereas S stores all objects as separate files in a directory somewhere (usually '`.Data`' under the current directory), R does not. All objects in R are stored internally. When R is started up it grabs a piece of memory and uses it to store the objects. R performs its own memory management of this piece of memory, growing and shrinking its size as needed. Having everything in memory is necessary because it is not really possible to externally maintain all relevant "environments" of symbol/value pairs. This difference also seems to make R *faster* than S.

The down side is that if R crashes you will lose all the work for the current session. Saving and restoring the memory "images" (the functions and data stored in R's internal memory at any time) can be a bit slow, especially if they are big. In S this does not happen, because

everything is saved in disk files and if you crash nothing is likely to happen to them. (In fact, one might conjecture that the S developers felt that the price of changing their approach to persistent storage just to accommodate lexical scope was far too expensive.) Hence, when doing important work, you might consider saving often (see Section 7.4 [How can I save my workspace?], page 45) to safeguard against possible crashes. Other possibilities are logging your sessions, or have your R commands stored in text files which can be read in using `source()`.

> **Note:** If you run R from within Emacs (see Chapter 6 [R and Emacs], page 43), you can save the contents of the interaction buffer to a file and conveniently manipulate it using `ess-transcript-mode`, as well as save source copies of all functions and data used.

3.3.2 Models

There are some differences in the modeling code, such as

- Whereas in S, you would use `lm(y ~ x^3)` to regress `y` on `x^3`, in R, you have to insulate powers of numeric vectors (using `I()`), i.e., you have to use `lm(y ~ I(x^3))`.
- The glm family objects are implemented differently in R and S. The same functionality is available but the components have different names.
- Option `na.action` is set to `"na.omit"` by default in R, but not set in S.
- Terms objects are stored differently. In S a terms object is an expression with attributes, in R it is a formula with attributes. The attributes have the same names but are mostly stored differently. The major difference in functionality is that a terms object is subscriptable in S but not in R. If you can't imagine why this would matter then you don't need to know.
- Finally, in R `y~x+0` is an alternative to `y~x-1` for specifying a model with no intercept. Models with no parameters at all can be specified by `y~0`.

3.3.3 Others

Apart from lexical scoping and its implications, R follows the S language definition in the Blue and White Books as much as possible, and hence really is an "implementation" of S. There are some intentional differences where the behavior of S is considered "not clean". In general, the rationale is that R should help you detect programming errors, while at the same time being as compatible as possible with S.

Some known differences are the following.

- In R, if `x` is a list, then `x[i] <- NULL` and `x[[i]] <- NULL` remove the specified elements from `x`. The first of these is incompatible with S, where it is a no-op. (Note that you can set elements to `NULL` using `x[i] <- list(NULL)`.)
- In S, the functions named `.First` and `.Last` in the '`.Data`' directory can be used for customizing, as they are executed at the very beginning and end of a session, respectively.

  In R, the startup mechanism is as follows. R first sources the system startup file '`$R_HOME/library/base/R/Rprofile`'. Then, it searches for a site-wide startup profile unless the command line option '`--no-site-file`' was given. The name of this file

is taken from the value of the `R_PROFILE` environment variable. If that variable is unset, the default is '`$R_HOME/etc/Rprofile.site`' ('`$R_HOME/etc/Rprofile`' in versions prior to 1.4.0). This code is loaded in package **base**. Then, unless '`--no-init-file`' was given, R searches for a file called '`.Rprofile`' in the current directory or in the user's home directory (in that order) and sources it into the user workspace. It then loads a saved image of the user workspace from '`.RData`' in case there is one (unless '`--no-restore`' was specified). If needed, the functions `.First()` and `.Last()` should be defined in the appropriate startup profiles.

- In R, `T` and `F` are just variables being set to `TRUE` and `FALSE`, respectively, but are not reserved words as in S and hence can be overwritten by the user. (This helps e.g. when you have factors with levels `"T"` or `"F"`.) Hence, when writing code you should always use `TRUE` and `FALSE`.
- In R, `dyn.load()` can only load *shared objects*, as created for example by *`R CMD SHLIB`*.
- In R, `attach()` currently only works for lists and data frames, but not for directories. (In fact, `attach()` also works for R data files created with `save()`, which is analogous to attaching directories in S.) Also, you cannot attach at position 1.
- Categories do not exist in R, and never will as they are deprecated now in S. Use factors instead.
- In R, `For()` loops are not necessary and hence not supported.
- In R, `assign()` uses the argument '`envir=`' rather than '`where=`' as in S.
- The random number generators are different, and the seeds have different length.
- R passes integer objects to C as `int *` rather than `long *` as in S.
- R has no single precision storage mode. However, as of version 0.65.1, there is a single precision interface to C/FORTRAN subroutines.
- By default, `ls()` returns the names of the objects in the current (under R) and global (under S) environment, respectively. For example, given

  ```
  x <- 1; fun <- function() {y <- 1; ls()}
  ```

  then `fun()` returns `"y"` in R and `"x"` (together with the rest of the global environment) in S.
- R allows for zero-extent matrices (and arrays, i.e., some elements of the `dim` attribute vector can be 0). This has been determined a useful feature as it helps reducing the need for special-case tests for empty subsets. For example, if `x` is a matrix, `x[, FALSE]` is not `NULL` but a "matrix" with 0 columns. Hence, such objects need to be tested for by checking whether their `length()` is zero (which works in both R and S), and not using `is.null()`.
- Named vectors are considered vectors in R but not in S (e.g., `is.vector(c(a = 1:3))` returns `FALSE` in S and `TRUE` in R).
- Data frames are not considered as matrices in R (i.e., if `DF` is a data frame, then `is.matrix(DF)` returns `FALSE` in R and `TRUE` in S).
- R by default uses treatment contrasts in the unordered case, whereas S uses the Helmert ones. This is a deliberate difference reflecting the opinion that treatment contrasts are more natural.

- In R, the argument of a replacement function which corresponds to the right hand side must be named 'value'. E.g., `f(a) <- b` is evaluated as `a <- "f<-"(a, value = b)`. S always takes the last argument, irrespective of its name.
- In S, `substitute()` searches for names for substitution in the given expression in three places: the actual and the default arguments of the matching call, and the local frame (in that order). R looks in the local frame only, with the special rule to use a "promise" if a variable is not evaluated. Since the local frame is initialized with the actual arguments or the default expressions, this is usually equivalent to S, until assignment takes place.
- In S, the index variable in a `for()` loop is local to the inside of the loop. In R it is local to the environment where the `for()` statement is executed.
- In S, `tapply(simplify=TRUE)` returns a vector where R returns a one-dimensional array (which can have named dimnames).
- In S(-PLUS) the C locale is used, whereas in R the current operating system locale is used for determining which characters are alphanumeric and how they are sorted. This affects the set of valid names for R objects (for example accented chars may be allowed in R) and ordering in sorts and comparisons (such as whether `"aA" < "Bb"` is true or false). From version 1.2.0 the locale can be (re-)set in R by the `Sys.setlocale()` function.
- In S, `missing(`*arg*`)` remains `TRUE` if *arg* is subsequently modified; in R it doesn't.
- From R version 1.3.0, `data.frame` strips `I()` when creating (column) names.
- In R, the string `"NA"` is not treated as a missing value in a character variable. Use `as.character(NA)` to create a missing character value.
- R disallows repeated formal arguments in function calls.

There are also differences which are not intentional, and result from missing or incorrect code in R. The developers would appreciate hearing about any deficiencies you may find (in a written report fully documenting the difference as you see it). Of course, it would be useful if you were to implement the change yourself and make sure it works.

3.4 Is there anything R can do that S-PLUS cannot?

Since almost anything you can do in R has source code that you could port to S-PLUS with little effort there will never be much you can do in R that you couldn't do in S-PLUS if you wanted to. (Note that using lexical scoping may simplify matters considerably, though.)

R offers several graphics features that S-PLUS does not, such as finer handling of line types, more convenient color handling (via palettes), gamma correction for color, and, most importantly, mathematical annotation in plot texts, via input expressions reminiscent of TEX constructs. See the help page for `plotmath`, which features an impressive on-line example. More details can be found in Paul Murrell and Ross Ihaka (2000), "An Approach to Providing Mathematical Annotation in Plots", *Journal of Computational and Graphical Statistics*, **9**, 582–599.

3.5 What is R-plus?

There is no such thing.

4 R Web Interfaces

Rweb is developed and maintained by Jeff Banfield. The Rweb Home Page provides access to all three versions of Rweb—a simple text entry form that returns output and graphs, a more sophisticated Javascript version that provides a multiple window environment, and a set of point and click modules that are useful for introductory statistics courses and require no knowledge of the R language. All of the Rweb versions can analyze Web accessible datasets if a URL is provided.

The paper "Rweb: Web-based Statistical Analysis", providing a detailed explanation of the different versions of Rweb and an overview of how Rweb works, was published in the Journal of Statistical Software (http://www.stat.ucla.edu/journals/jss/v04/i01/).

Ulf Bartel is working on **R-Online**, a simple on-line programming environment for R which intends to make the first steps in statistical programming with R (especially with time series) as easy as possible. There is no need for a local installation since the only requirement for the user is a JavaScript capable browser. See http://osvisions.com/r-online/ for more information.

CGIwithR is an R add-on package by David Firth and has its home page at http://www.stats.ox.ac.uk/~firth/CGIwithR/index.html. It provides some simple extensions to R to facilitate running R scripts through the CGI interface to a web server. It is easily installed using Apache under Linux and in principle should run on any platform that supports R and a web server provided that the installer has the necessary security permissions.

Rcgi is a CGI WWW interface to R by Mark J. Ray. It had the ability to use "embedded code": you could mix user input and code, allowing the HTML author to do anything from load in data sets to enter most of the commands for users without writing CGI scripts. Graphical output was possible in PostScript or GIF formats and the executed code was presented to the user for revision. However, it is not clear if the project is still active and recent attempts to contact the author or find the package have failed.

5 R Add-On Packages

5.1 Which add-on packages exist for R?

5.1.1 Add-on packages in R

The R distribution comes with the following extra packages:

ctest A collection of Classical TESTs, including the Ansari-Bradley, Bartlett, chi-squared, Fisher, Kruskal-Wallis, Kolmogorov-Smirnov, t, and Wilcoxon tests.

eda Exploratory Data Analysis. Currently only contains functions for robust line fitting, and median polish and smoothing.

grid A rewrite of the graphics layout capabilities, plus some support for interaction. (Added in R 1.8.0).

lqs Resistant regression and covariance estimation.

methods Formally defined methods and classes for R objects, plus other programming tools, as described in the Green Book.

mle Generic (smooth) likelihood maximization and profiling. (Added in R 1.8.0).

modreg MODern REGression: smoothing and local methods.

mva MultiVariate Analysis. Currently contains code for principal components, canonical correlations, metric multidimensional scaling, factor analysis, and hierarchical and k-means clustering.

nls Nonlinear regression routines.

splines Regression spline functions and classes.

stepfun Code for dealing with STEP FUNctions, including empirical cumulative distribution functions.

tcltk Interface and language bindings to Tcl/Tk GUI elements.

tools Tools for package development and administration.

ts Time Series.

5.1.2 Add-on packages from CRAN

The following packages are available from the CRAN '`src/contrib`' area. (Packages denoted as *Recommended* are to be included in all binary distributions of R.)

AnalyzeFMRI
Functions for I/O, visualisation and analysis of functional Magnetic Resonance Imaging (fMRI) datasets stored in the ANALYZE format.

Bhat Functions for general likelihood exploration (MLE, MCMC, CIs).

CDNmoney Components of Canadian monetary aggregates.

CGIwithR Facilities for the use of R to write CGI scripts.

CircStats Circular Statistics, from "Topics in Circular Statistics" by S. Rao Jammalamadaka and A. SenGupta, 2001, World Scientific.

CoCoAn Constrained Correspondence Analysis.

DAAG Various data sets used in examples and exercises in "Data Analysis and Graphics Using R" by John H. Maindonald and W. John Brown, 2003.

DBI A common database interface (DBI) class and method definitions. All classes in this package are virtual and need to be extended by the various DBMS implementations.

Davies Functions for the Davies quantile function and the Generalized Lambda distribution.

Design Regression modeling, testing, estimation, validation, graphics, prediction, and typesetting by storing enhanced model design attributes in the fit. Design is a collection of about 180 functions that assist and streamline modeling, especially for biostatistical and epidemiologic applications. It also contains new functions for binary and ordinal logistic regression models and the Buckley-James multiple regression model for right-censored responses, and implements penalized maximum likelihood estimation for logistic and ordinary linear models. Design works with almost any regression model, but it was especially written to work with logistic regression, Cox regression, accelerated failure time models, ordinary linear models, and the Buckley-James model.

Devore5 Data sets and sample analyses from "Probability and Statistics for Engineering and the Sciences (5th ed)" by Jay L. Devore, 2000, Duxbury.

Devore6 Data sets and sample analyses from "Probability and Statistics for Engineering and the Sciences (6th ed)" by Jay L. Devore, 2003, Duxbury.

EMV Estimation of missing values in a matrix by a k-th nearest neighboors algorithm.

GRASS An interface between the GRASS geographical information system and R, based on starting R from within the GRASS environment and chosen LOCATION_NAME and MAPSET. Wrapper and helper functions are provided for a range of R functions to match the interface metadata structures.

GenKern Functions for generating and manipulating generalised binned kernel density estimates.

HI Simulation from distributions supported by nested hyperplanes.

Hmisc Functions useful for data analysis, high-level graphics, utility operations, functions for computing sample size and power, importing datasets, imputing missing values, advanced table making, variable clustering, character string manipulation, conversion of S objects to LaTeX code, recoding variables, and bootstrap repeated measures analysis.

HyperbolicDist
Basic functions for the hyperbolic distribution: probability density function, distribution function, quantile function, a routine for generating observations from the hyperbolic, and a function for fitting the hyperbolic distribution to data.

ISwR Data sets for "Introductory Statistics with R" by Peter Dalgaard, 2002, Springer.

KMsurv Data sets and functions for "Survival Analysis, Techniques for Censored and Truncated Data" by Klein and Moeschberger, 1997, Springer.

KernSmooth
Functions for kernel smoothing (and density estimation) corresponding to the book "Kernel Smoothing" by M. P. Wand and M. C. Jones, 1995. *Recommended.*

MASS Functions and datasets from the main package of Venables and Ripley, "Modern Applied Statistics with S". Contained in the '`VR`' bundle. *Recommended.*

MCMCpack
Markov chain Monte Carlo (MCMC) package: functions for posterior simulation for a number of statistical models.

MPV Data sets from the book "Introduction to Linear Regression Analysis" by D. C. Montgomery, E. A. Peck, and C. G. Vining, 2001, John Wiley and Sons.

Matrix A Matrix package.

NISTnls A set of test nonlinear least squares examples from NIST, the U.S. National Institute for Standards and Technology.

Oarray Arrays with arbitrary offsets.

PHYLOGR
Manipulation and analysis of phylogenetically simulated data sets (as obtained from PDSIMUL in package PDAP) and phylogenetically-based analyses using GLS.

PTAk A multiway method to decompose a tensor (array) of any order, as a generalisation of SVD also supporting non-identity metrics and penalisations. Also includes some other multiway methods.

R2HTML Functions for exporting R objects & graphics in an HTML document.

RArcInfo Functions to import Arc/Info V7.x coverages and data.

RColorBrewer
ColorBrewer palettes for drawing nice maps shaded according to a variable.

RMySQL An interface between R and the MySQL database system.

RODBC An ODBC database interface.

ROracle Oracle Database Interface driver for R. Uses the ProC/C++ embedded SQL.

RQuantLib
Provides access to (some) of the QuantLib functions from within R; currently limited to some Option pricing and analysis functions. The QuantLib project aims to provide a comprehensive software framework for quantitative finance.

RSQLite Database Interface R driver for SQLite. Embeds the SQLite database engine in R.

RSvgDevice
A graphics device for R that uses the new w3.org XML standard for Scalable Vector Graphics.

RadioSonde
A collection of programs for reading and plotting SKEW-T,log p diagrams and wind profiles for data collected by radiosondes (the typical weather balloon-borne instrument).

RandomFields
Creating random fields using various methods.

Rcmdr A platform-independent basic-statistics GUI (graphical user interface) for R, based on the **tcltk** package.

RmSQL An interface between R and the mSQL database system.

Rwave An environment for the time-frequency analysis of 1-D signals (and especially for the wavelet and Gabor transforms of noisy signals), based on the book "Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S" by Rene Carmona, Wen L. Hwang and Bruno Torresani, 1998, Academic Press.

SASmixed Data sets and sample linear mixed effects analyses corresponding to the examples in "SAS System for Mixed Models" by R. C. Littell, G. A. Milliken, W. W. Stroup and R. D. Wolfinger, 1996, SAS Institute.

SenSrivastava
Collection of datasets from "Regression Analysis, Theory, Methods and Applications" by A. Sen and M. Srivastava, 1990, Springer-Verlag.

SparseM Basic linear algebra for sparse matrices.

StatDataML
Read and write StatDataML.

SuppDists Ten distributions supplementing those built into R (Inverse Gauss, Kruskal-Wallis, Kendall's Tau, Friedman's chi squared, Spearman's rho, maximum F ratio, the Pearson product moment correlation coefficiant, Johnson distributions, normal scores and generalized hypergeometric distributions).

VLMC Functions, classes & methods for estimation, prediction, and simulation (bootstrap) of VLMC (Variable Length Markov Chain) models.

VaR Methods for calculation of Value at Risk (VaR).

XML Facilities for reading XML documents and DTDs.

abind Combine multi-dimensional arrays.

acepack ACE (Alternating Conditional Expectations) and AVAS (Additivity and VAriance Stabilization for regression) methods for selecting regression transformations.

adapt Adaptive quadrature in up to 20 dimensions.

ade4 Multivariate data analysis and graphical display.

agce Analysis of growth curve experiments.

akima Linear or cubic spline interpolation for irregularly gridded data.

amap Another Multidimensional Analysis Package.

anm Analog model for statistical/empirical downscaling.

ape Analyses of Phylogenetics and Evolution, providing functions for reading and plotting phylogenetic trees in parenthetic format (standard Newick format), analyses of comparative data in a phylogenetic framework, analyses of diversification and macroevolution, computing distances from allelic and nucleotide data, reading nucleotide sequences from GenBank via internet, and several tools such as Mantel's test, computation of minimum spanning tree, or the population parameter theta based on various approaches.

ash David Scott's ASH routines for 1D and 2D density estimation.

asypow A set of routines that calculate power and related quantities utilizing asymptotic likelihood ratio methods.

aws Functions to perform adaptive weights smoothing.

bim Bayesian interval mapping diagnostics: functions to interpret QTLCart and Bmapqtl samples.

bindata Generation of correlated artificial binary data.

blighty Function for drawing the coastline of the United Kingdom.

boolean Boolean logit and probit: a procedure for testing Boolean hypotheses.

boot Functions and datasets for bootstrapping from the book "Bootstrap Methods and Their Applications" by A. C. Davison and D. V. Hinkley, 1997, Cambridge University Press. *Recommended*.

bootstrap Software (bootstrap, cross-validation, jackknife), data and errata for the book "An Introduction to the Bootstrap" by B. Efron and R. Tibshirani, 1993, Chapman and Hall.

bqtl QTL mapping toolkit for inbred crosses and recombinant inbred lines. Includes maximum likelihood and Bayesian tools.

brlr Bias-reduced logistic regression: fits logistic regression models by maximum penalized likelihood.

car Companion to Applied Regression, containing functions for applied regession, linear models, and generalized linear models, with an emphasis on regression diagnostics, particularly graphical diagnostic methods.

cat Analysis of categorical-variable datasets with missing values.

cclust Convex clustering methods, including k-means algorithm, on-line update algorithm (Hard Competitive Learning) and Neural Gas algorithm (Soft Competitive Learning) and calculation of several indexes for finding the number of clusters in a data set.

cfa Analysis of configuration frequencies.

chron A package for working with chronological objects (times and dates).

class Functions for classification (k-nearest neighbor and LVQ). Contained in the '`VR`' bundle. *Recommended*.

classPP Projection Pursuit for supervised classification.

clim.pact Climate analysis and downscaling for monthly and daily data.

clines Calculates Contour Lines.

cluster Functions for cluster analysis. *Recommended*.

cmprsk Estimation, testing and regression modeling of subdistribution functions in competing risks.

cobs Constrained B-splines: qualitatively constrained (regression) smoothing via linear programming.

coda Output analysis and diagnostics for Markov Chain Monte Carlo (MCMC) simulations.

combinat Combinatorics utilities.

conf.design A series of simple tools for constructing and manipulating confounded and fractional factorial designs.

cramer Routine for the multivariate nonparametric Cramer test.

date Functions for dealing with dates. The most useful of them accepts a vector of input dates in any of the forms '`8/30/53`', '`30Aug53`', '`30 August 1953`', ..., '`August 30 53`', or any mixture of these.

dblcens Calculates the NPMLE of the survival distribution for doubly censored data.

deal Bayesian networks with continuous and/or discrete variables can be learned and compared from data.

deldir Calculates the Delaunay triangulation and the Dirichlet or Voronoi tesselation (with respect to the entire plane) of a planar point set.

diamonds Functions for illustrating aperture-4 diamond partitions in the plane, or on the surface of an octahedron or icosahedron, for use as analysis or sampling grids.

dichromat Color schemes for dichromats: collapse red-green distinctions to simulate the effects of colour-blindness.

diptest Compute Hartigan's dip test statistic for unimodality.

dispmod Functions for modelling dispersion in GLMs.

dr Functions, methods, and datasets for fitting dimension reduction regression, including pHd and inverse regression methods SIR and SAVE.

dse Dynamic System Estimation, a multivariate time series package. Contains **dse1** (the base system, including multivariate ARMA and state space models), **dse2** (extensions for evaluating estimation techniques, forecasting, and for evaluating forecasting model), **tframe** (functions for writing code that is independent of the representation of time). and **setRNG** (a mechanism for generating the same random numbers in S and R).

e1071 Miscellaneous functions used at the Department of Statistics at TU Wien (E1071), including moments, short-time Fourier transforms, Independent Component Analysis, Latent Class Analysis, support vector machines, and fuzzy clustering, shortest path computation, bagged clustering, and some more.

effects Graphical and tabular effect displays, e.g., of interactions, for linear and generalised linear models.

eha A package for survival and event history analysis.

ellipse Package for drawing ellipses and ellipse-like confidence regions.

emme2 Functions to read from and write to an EMME/2 databank.

emplik Empirical likelihood ratio for means/quantiles/hazards from possibly right censored data.

evd Functions for extreme value distributions. Extends simulation, distribution, quantile and density functions to univariate, bivariate and (for simulation) multivariate parametric extreme value distributions, and provides fitting functions which calculate maximum likelihood estimates for univariate and bivariate models.

exactLoglinTest Monte Carlo exact tests for log-linear models.

exactRankTests Computes exact p-values and quantiles using an implementation of the Streitberg/Roehmel shift algorithm.

fastICA Implementation of FastICA algorithm to perform Independent Component Analysis (ICA) and Projection Pursuit.

fdim Functions for calculating fractal dimension.

fields A collection of programs for curve and function fitting with an emphasis on spatial data. The major methods implemented include cubic and thin plate splines, universal Kriging and Kriging for large data sets. The main feature is that any covariance function implemented in R can be used for spatial prediction.

flexmix Flexible Mixture Modeling: a general framework for finite mixtures of regression models using the EM algorithm.

foreign Functions for reading and writing data stored by statistical software like Minitab, SAS, SPSS, Stata, etc. *Recommended.*

forward Forward search approach to robust analysis in linear and generalized linear regression models.

fpc Fixed point clusters, clusterwise regression and discriminant plots.

fracdiff Maximum likelihood estimation of the parameters of a fractionally differenced ARIMA(p, d, q) model (Haslett and Raftery, Applied Statistics, 1989).

ftnonpar Features and strings for nonparametric regression.

g.data Create and maintain delayed-data packages (DDP's).

gafit Genetic algorithm for curve fitting.

gbm Generalized Boosted Regression Models: implements extensions to Freund and Schapire's AdaBoost algorithm and J. Friedman's gradient boosting machine. Includes regression methods for least squares, absolute loss, logistic, Poisson, Cox proportional hazards partial likelihood, and AdaBoost exponential loss.

gee An implementation of the Liang/Zeger generalized estimating equation approach to GLMs for dependent data.

geepack Generalized estimating equations solver for parameters in mean, scale, and correlation structures, through mean link, scale link, and correlation link. Can also handle clustered categorical responses.

genetics Classes and methods for handling genetic data. Includes classes to represent genotypes and haplotypes at single markers up to multiple markers on multiple chromosomes, and functions for allele frequencies, flagging homo/heterozygotes, flagging carriers of certain alleles, computing disequlibrium, testing Hardy-Weinberg equilibrium, . . .

geoR Functions to perform geostatistical data analysis including model-based methods.

geoRglm Functions for inference in generalised linear spatial models.

ggm Functions for defining directed acyclic graphs and undirected graphs, finding induced graphs and fitting Gaussian Markov models.

gld Basic functions for the generalised (Tukey) lambda distribution.

glmmML A Maximum Likelihood approach to generalized linear models with random intercept.

gpclib General polygon clipping routines for R based on Alan Murta's C library.

grasper Generalized Regression Analysis and Spatial Predictions for R.

gregmisc Miscellaneous functions written/maintained by Gregory R. Warnes.

gridBase Integration of base and grid graphics.

gss A comprehensive package for structural multivariate function estimation using smoothing splines.

gstat multivariable geostatistical modelling, prediction and simulation. Includes code for variogram modelling; simple, ordinary and universal point or block (co)kriging, sequential Gaussian or indicator (co)simulation, and map plotting functions.

gtkDevice GTK graphics device driver that may be used independently of the R-GNOME interface and can be used to create R devices as embedded components in a GUI using a Gtk drawing area widget, e.g., using RGtk.

haplo.score
Score tests for association of traits with haplotypes when linkage phase is ambiguous.

hdf5 Interface to the NCSA HDF5 library.

hier.part Hierarchical Partitioning: variance partition of a multivariate data set.

homals Homogeneity Analysis (HOMALS) package with optional Tcl/Tk interface.

hwde Models and tests for departure from Hardy-Weinberg equilibrium and independence between loci.

ifs Iterated Function Systems distribution function estimator.

ineq Inequality, concentration and poverty measures, and Lorenz curves (empirical and theoretic).

ipred Improved predictive models by direct and indirect bootstrap aggregation in classification and regression as well as resampling based estimators of prediction error.

ismev Functions to support the computations carried out in "An Introduction to Statistical Modeling of Extreme Values;' by S. Coles, 2001, Springer. The functions may be divided into the following groups; maxima/minima, order statistics, peaks over thresholds and point processes.

its An S4 class for handling irregular time series.

knnTree Construct or predict with k-nearest-neighbor classifiers, using cross-validation to select k, choose variables (by forward or backwards selection), and choose scaling (from among no scaling, scaling each column by its SD, or scaling each column by its MAD). The finished classifier will consist of a classification tree with one such k-nn classifier in each leaf.

lars Least Angle Regression, Lasso and Forward Stagewise: efficient procedures for fitting an entire lasso sequence with the cost of a single least squares fit.

lasso2 Routines and documentation for solving regression problems while imposing an L1 constraint on the estimates, based on the algorithm of Osborne et al. (1998)

lattice Lattice graphics, an implementation of Trellis Graphics functions. *Recommended.*

leaps A package which performs an exhaustive search for the best subsets of a given set of potential regressors, using a branch-and-bound algorithm, and also performs searches using a number of less time-consuming techniques.

lgtdl A set of methods for longitudinal data objects.

linprog Solve linear programming/linear optimization problems by using the simplex algorithm.

lme4 Fit linear and generalized linear mixed-effects models.

lmeSplines
Fit smoothing spline terms in Gaussian linear and nonlinear mixed-effects models.

lmm Linear mixed models.

lmtest A collection of tests on the assumptions of linear regression models from the book "The linear regression model under test" by W. Kraemer and H. Sonnberger, 1986, Physica.

locfit Local Regression, likelihood and density estimation.

logistf Firth's bias reduced logistic regression approach with penalized profile likelihood based confidence intervals for parameter estimates.

logspline Logspline density estimation.

lokern Kernel regression smoothing with adaptive local or global plug-in bandwidth selection.

lpSolve Functions that solve general linear/integer problems, assignment problems, and transportation problems via interfacing Lp_solve.

lpridge Local polynomial (ridge) regression.

mapdata Supplement to package **maps**, providing the larger and/or higher-resolution databases.

mapproj Map Projections: converts latitude/longitude into projected coordinates.

maps Draw geographical maps. Projection code and larger maps are in separate packages.

maptools Set of tools for manipulating and reading geographic data, in particular ESRI shapefiles.

maptree Functions with example data for graphing and mapping models from hierarchical clustering and classification and regression trees.

maxstat Maximally selected rank and Gauss statistics with several p-value approximations.

mclust Model-based cluster analysis: the 2002 version of MCLUST.

mclust1998
Model-based cluster analysis: the 1998 version of MCLUST.

mda Code for mixture discriminant analysis (MDA), flexible discriminant analysis (FDA), penalized discriminant analysis (PDA), multivariate additive regression splines (MARS), adaptive back-fitting splines (BRUTO), and penalized regression.

merror Accuracy and precision of measurements.

mgcv Routines for GAMs and other genralized ridge regression problems with multiple smoothing parameter selection by GCV or UBRE. *Recommended.*

mimR An R interface to MIM for graphical modeling in R.

mix Estimation/multiple imputation programs for mixed categorical and continuous data.

mlbench A collection of artificial and real-world machine learning benchmark problems, including the Boston housing data.

mmlcr Mixed-mode latent class regression (also known as mixed-mode mixture model regression or mixed-mode mixture regression models) which can handle both longitudinal and one-time responses.

moc Fits a variety of mixtures models for multivariate observations with user-difined distributions and curves.

msm Functions for fitting continuous-time Markov multi-state models to categorical processes observed at arbitrary times, optionally with misclassified responses, and covariates on transition or misclassification rates.

muhaz Hazard function estimation in survival analysis.

multcomp Multiple comparison procedures for the one-way layout.

multidim Multidimensional descriptive statistics: factorial methods and classification.

multiv Functions for hierarchical clustering, partitioning, bond energy algorithm, Sammon mapping, PCA and correspondence analysis.

mvnmle ML estimation for multivariate normal data with missing values.

mvnormtest Generalization of the Shapiro-Wilk test for multivariate variables.

mvtnorm Multivariate normal and t distributions.

ncdf Interface to Unidata netCDF data files.

ncomplete Functions to perform the regression depth method (RDM) to binary regression to approximate the minimum number of observations that can be removed such that the reduced data set has complete separation.

negenes Estimating the number of essential genes in a genome on the basis of data from a random transposon mutagenesis experiment, through the use of a Gibbs sampler.

netCDF Read data from netCDF files.

nlme Fit and compare Gaussian linear and nonlinear mixed-effects models. *Recommended.*

nlmeODE Combine the **nlme** and **odesolve** packages for mixed-effects modelling using differential equations.

nlrq Nonlinear quantile regression.

nnet Software for single hidden layer perceptrons ("feed-forward neural networks"), and for multinomial log-linear models. Contained in the '`VR`' bundle. *Recommended.*

nor1mix One-dimensional normal mixture models classes, for, e.g., density estimation or clustering algorithms research and teaching; providing the widely used Marron-Wand densities.

norm Analysis of multivariate normal datasets with missing values.

normalp A collection of utilities for normal of order p distributions (General Error Distributions).

nortest Five omnibus tests for the composite hypothesis of normality.

noverlap Functions to perform the regression depth method (RDM) to binary regression to approximate the amount of overlap, i.e., the minimal number of observations that need to be removed such that the reduced data set has no longer overlap.

npmc Nonparametric Multiple Comparisons: provides simultaneous rank test procedures for the one-way layout without presuming a certain distribution.

nprq Nonparametric and sparse quantile regression methods.

odesolve An interface for the Ordinary Differential Equation (ODE) solver lsoda. ODEs are expressed as R functions.

orientlib Representations, conversions and display of orientation SO(3) data.

oz Functions for plotting Australia's coastline and state boundaries.

pamr Pam: Prediction Analysis for Microarrays.

pan Multiple imputation for multivariate panel or clustered data.

panel Functions and datasets for fitting models to Panel data.

pastecs Package for Analysis of Space-Time Ecological Series.

pcurve Fits a principal curve to a numeric multivariate dataset in arbitrary dimensions. Produces diagnostic plots. Also calculates Bray-Curtis and other distance matrices and performs multi-dimensional scaling and principal component analyses.

pear Periodic Autoregression Analysis.

permax Functions intended to facilitate certain basic analyses of DNA array data, especially with regard to comparing expression levels between two types of tissue.

pinktoe Converts S trees to HTML/Perl files for interactive tree traversal.

pixmap Functions for import, export, plotting and other manipulations of bitmapped images.

pls.pcr Multivariate regression by PLS and PCR.

polspline Routines for the polynomial spline fitting routines hazard regression, hazard estimation with flexible tails, logspline, lspec, polyclass, and polymars, by C. Kooperberg and co-authors.

polynom A collection of functions to implement a class for univariate polynomial manipulations.

pps Functions to select samples using PPS (probability proportional to size) sampling, for stratified simple random sampling, and to compute joint inclusion probabilities for Sampford's method of PPS sampling.

prabclus Distance based parametric bootstrap tests for clustering, mainly thought for presence-absence data (clustering of species distribution maps). Jaccard and Kulczynski distance measures, clustering of MDS scores, and nearest neighbor based noise detection.

princurve Fits a principal curve to a matrix of points in arbitrary dimension.

pspline Smoothing splines with penalties on order m derivatives.

psy Various procedures used in psychometry: Kappa, ICC, Cronbach alpha, screeplot, PCA and related methods.

qtl Analysis of experimental crosses to identify QTLs.

quadprog For solving quadratic programming problems.

quantreg Quantile regression and related methods.

qvcalc Functions to compute quasi-variances and associated measures of approximation error.

randomForest
Breiman's random forest classifier.

relimp Functions to facilitate inference on the relative importance of predictors in a linear or generalized linear model.

rgenoud R version of GENetic Optimization Using Derivatives.

rimage Functions for image processing, including Sobel filter, rank filters, fft, histogram equalization, and reading JPEG files.

rmeta Functions for simple fixed and random effects meta-analysis for two-sample comparison of binary outcomes.

rpart Recursive PARTitioning and regression trees. *Recommended*.

rpvm R interface to PVM (Parallel Virtual Machine). Provides interface to PVM APIs, and examples and documentation for its use.

rqmcmb2 Markov chain marginal bootstrap for quantile regression.

rsprng Provides interface to SPRNG (Scalable Parallel Random Number Generators) APIs, and examples and documentation for its use.

sampfling Implements a modified version of the Sampford sampling algorithm. Given a quantity assigned to each unit in the population, samples are drawn with probability proportional to te product of the quantities of the units included in the sample.

sca Simple Component Analysis.

scatterplot3d
Plots a three dimensional (3D) point cloud perspectively.

seacarb Calculates parameters of the seawater carbonate system.

seao Simple Evolutionary Algorithm Optimization.

seao.gui Simple Evolutionary Algorithm Optimization: graphical user interface.

segmented Functions to estimate break-points of segmented relationships in regression models (GLMs).

sem Functions for fitting general linear Structural Equation Models (with observed and unobserved variables) by the method of maximum likelihood using the RAM approach.

serialize Simple interfce for serializing to connections.

session Functions for interacting with, saving and restoring R sessions.

sgeostat An object-oriented framework for geostatistical modeling.

shapefiles Functions to read and write ESRI shapefiles.

shapes Routines for the statistical analysis of shapes, including procrustes analysis, displaying shapes and principal components, testing for mean shape difference, thin-plate spline transformation grids and edge superimposition methods.

simpleboot Simple bootstrap routines.

sm Software linked to the book "Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-PLUS Illustrations" by A. W. Bowman and A. Azzalini (1997), Oxford University Press.

sma Functions for exploratory (statistical) microarray analysis.

smoothSurv Survival regression with smoothed error distribution.

sn Functions for manipulating skew-normal probability distributions and for fitting them to data, in the scalar and the multivariate case.

sna A range of tools for social network analysis, including node and graph-level indices, structural distance and covariance methods, structural equivalence detection, p* modeling, and network visualization.

snow Simple Network of Workstations: support for simple parallel computing in R.

som Self-Organizing Maps (with application in gene clustering).

sound A sound interface for R: Basic functions for dealing with '`.wav`' files and sound samples.

spatial Functions for kriging and point pattern analysis from "Modern Applied Statistics with S" by W. Venables and B. Ripley. Contained in the '`VR`' bundle. *Recommended*.

spatstat Data analysis and modelling of two-dimensional point patterns, including multitype points and spatial covariates.

spdep A collection of functions to create spatial weights matrix objects from polygon contiguities, from point patterns by distance and tesselations, for summarising these objects, and for permitting their use in spatial data analysis; a collection of tests for spatial autocorrelation, including global Moran's I and Geary's C, local Moran's I, saddlepoint approximations for global and local Moran's I; and functions for estimating spatial simultaneous autoregressive (SAR) models. (Was formerly the three packages: **spweights**, **sptests**, and **spsarlm**.)

splancs Spatial and space-time point pattern analysis functions.

statmod Miscellaneous biostatistical modelling functions.

strucchange Various tests on structural change in linear regression models.

subselect A collection of functions which assess the quality of variable subsets as surrogates for a full data set, and search for subsets which are optimal under various criteria.

survey Summary statistics, generalized linear models, and general maximum likelihood estimation for stratified, cluster-sampled, unequally weighted survey samples.

survival Functions for survival analysis, including penalised likelihood. *Recommended.*

survrec Survival analysis for recurrent event data.

systemfit Contains functions for fitting simultaneous systems of equations using Ordinary Least Sqaures (OLS), Two-Stage Least Squares (2SLS), and Three-Stage Least Squares (3SLS).

tapiR Tools for accessing (UK) parliamentary information in R.

tensor Tensor product of arrays.

tkrplot Simple mechanism for placing R graphics in a Tk widget.

tree Classification and regression trees.

tripack A constrained two-dimensional Delaunay triangulation package.

tseries Package for time series analysis with emphasis on non-linear modelling.

udunits Interface to Unidata's routines to convert units.

vardiag Interactive variogram diagnostics.

vcd Functions and data sets based on the book "Visualizing Categorical Data" by Michael Friendly.

vegan Various help functions for vegetation scientists and community ecologists.

waveslim Basic wavelet routines for time series analysis.

wavethresh Software to perform 1-d and 2-d wavelet statistics and transforms.

wle Robust statistical inference via a weighted likelihood approach.

xgobi Interface to the XGobi and XGvis programs for graphical data analysis.

xtable Export data to LaTeX and HTML tables.

See CRAN '`src/contrib/PACKAGES`' for more information.

There is also a CRAN '`src/contrib/Devel`' directory which contains packages still "under development" or depending on features only present in the current development versions of R. Volunteers are invited to give these a try, of course. This area of CRAN currently contains

Dopt Finding D-optimal experimental designs.

GLMMGibbs
Generalised Linear Mixed Models by Gibbs sampling.

RPgSQL Provides methods for accessing data stored in PostgreSQL tables.

Rmpi An interface (wrapper) to MPI (Message-Passing Interface) APIs. It also provides interactive R slave functionalities to make MPI programming easier in R than in C(++) or FORTRAN.

dseplus Extensions to **dse**, the Dynamic Systems Estimation multivariate time series package. Contains PADI, juice and monitoring extensions.

ensemble Ensembles of tree classifiers.

gllm Routines for log-linear models of incomplete contingency tables, including some latent class models via EM and Fisher scoring approaches.

meanscore Mean Score method for missing covariate data in logistic regression models.

runStat Running median and mean.

twostage Functions for optimal design of two-stage-studies using the Mean Score method.

write.snns Function for writing a SNNS pattern file from a data frame or matrix.

5.1.3 Add-on packages from Omegahat

The '`src/contrib/Omegahat`' Directory of a CRAN site contains yet unreleased packages from the Omegahat Project for Statistical Computing. Currently, there are

CORBA Dynamic CORBA client/server facilities for R. Connects to other CORBA-aware applications developed in arbitrary languages, on different machines and allows R functionality to be exported in the same way to other applications.

OOP OOP style classes and methods for R and S-PLUS. Object references and class-based method definition are supported in the style of languages such as Java and C++.

REmbeddedPostgres
Allows R functions and objects to be used to implement SQL functions — per-record, aggregate and trigger functions.

REventLoop
An abstract event loop mechanism that is toolkit independent and can be used to to replace the R event loop.

RGdkPixbuf
: S language functions to access the facilities in the GdkPixbuf library for manipulating images.

RGnumeric
: A plugin for the Gnumeric spreadsheet that allows R functions to be called from cells within the sheet, automatic recalculation, etc.

RGtk
: Facilities in the S language for programming graphical interfaces using Gtk, the Gnome GUI toolkit.

RGtkBindingGenerator
: A meta-package which generates C and R code to provide bindings to a Gtk-based library.

RGtkExtra
: A collection of S functions that provide an interface to the widgets in the gtk+extra library such as the GtkSheet data-grid display, icon list, file list and directory tree.

RGtkGlade
: S language bindings providing an interface to Glade, the interactive Gnome GUI creator.

RGtkHTML
: A collection of S functions that provide an interface to creating and controlling an HTML widget which can be used to display HTML documents from files or content generated dynamically in S.

RGtkViewers
: A collection of tools for viewing different S objects, databases, class and widget hierarchies, S source file contents, etc.

RJavaDevice
: A graphics device for R that uses Java components and graphics. APIs.

RObjectTables
: The C and S code allows one to define R objects to be used as elements of the search path with their own semantics and facilities for reading and writing variables. The objects implement a simple interface via R functions (either methods or closures) and can access external data, e.g., in other applications, languages, formats, . . .

RSMethods
: An implementation of S version 4 methods and classes for R, consistent with the basic material in "Programming with Data" by John M. Chambers, 1998, Springer NY.

RSPerl
: An interface from R to an embedded, persistent Perl interpreter, allowing one to call arbitrary Perl subroutines, classes and methods.

RSPython
: Allows Python programs to invoke S functions, methods, etc., and S code to call Python functionality.

RXLisp An interface to call XLisp-Stat functions from within R.

SASXML Example for reading XML files in SAS 8.2 manner.

SJava An interface from R to Java to create and call Java objects and methods.

SLanguage
Functions and C support utilities to support S language programming that can work in both R and S-PLUS.

SNetscape Plugin for Netscape and JavaScript.

SWinRegistry
Provides access from within R to read and write the Windows registry.

SWinTypeLibs
Provides ways to extract type information from type libraries and/or DCOM objects that describes the methods, properties, etc. of an interface.

SXalan Process XML documents using XSL functions implemented in R and dynamically substituting output from R.

Slcc Parses C source code, allowing one to analyze and automatically generate interfaces from S to that code, including the table of S-accessible native symbols, parameter count and type information, S constructors from C objects, call graphs, etc.

Sxslt An extension module for libxslt, the XML-XSL document translator, that allows XSL functions to be implemented via R functions.

5.1.4 Add-on packages from BioConductor

The Bioconductor Project produces an open source software framework that will assist biologists and statisticians working in bioinformatics, with primary emphasis on inference using DNA microarrays. The following R packages are contained in the current release of BioConductor, with more packages under development.

AnnBuilder
Assemble and process genomic annotation data, from databases such as GenBank, the Gene Ontology Consortium, LocusLink, UniGene, the UCSC Human Genome Project.

Biobase Object-oriented representation and manipulation of genomic data (S4 class structure).

DynDoc Functionality to create and interact with dynamic documents, vignettes, and other navigable documents.

MAGEML
Functionality to handle MAGEML documents.

RBGL An interface between the graph package and the Boost graph libraries, allowing for fast manipulation of graph objects in R.

ROC Receiver Operating Characteristic (ROC) approach for identifying genes that are differentially expressed in two types of samples.

RdbiPgSQL
: Methods for accessing data stored in PostgreSQL tables.

Rgraphviz An interface with Graphviz for plotting graph objects in R.

Ruuid Creates Universally Unique ID values (UUIDs) in R.

SAGElyzer
: Locates genes based on SAGE tags.

SNPtools Rudimentary structures for SNP data.

affy Methods for Affymetrix Oligonucleotide Arrays.

affyPLM For fitting Probe Level Models.

affycomp Graphics toolbox for assessment of Affymetrix expression measures.

affydata Affymetrix data for demonstration purposes.

annaffy Functions for handling data from Bioconductor Affymetrix annotation data packages.

annotate Associate experimental data in real time to biological metadata from web databases such as GenBank, LocusLink and PubMed. Process and store query results. Generate HTML reports of analyses.

ctc Tools to export and import Tree and Cluster to other programs.

daMA Functions for the efficient design of factorial two-color microarray experiments and for the statistical analysis of factorial microarray data.

edd Expression density diagnostics: graphical methods and pattern recognition algorithms for distribution shape classification.

externalVector
: Basic class definitions and generics for external pointer based vector objects for R.

factDesign A set of tools for analyzing data from factorial designed micraorray experiments. The functions can be used to evaluate appropriate tests of contrast and perform single outlier detection.

gcrma Background adjustment using sequence information.

genefilter Tools for sequentially filtering genes using a wide variety of filtering functions. Example of filters include: number of missing value, coefficient of variation of expression measures, ANOVA p-value, Cox model p-values. Sequential application of filtering functions to genes.

geneplotter
: Graphical tools for genomic data, for example for plotting expression data along a chromosome or producing color images of expression data matrices.

globaltest Testing globally whether a group of genes is significantly related to some clinical variable of interest.

gpls Classification using generalized partial least squares for two-group and multi-group classification.

graph Classes and tools for creating and manipulating graphs within R.

hexbin Binning functions, in particular hexagonal bins for graphing.

limma Linear models for microarray data.

makecdfenv
Two functions. One reads a Affymetrix chip description file (CDF) and creates a hash table environment containing the location/probe set membership mapping. The other creates a package that automatically loads that environment.

marrayClasses
Class definitions for pre-normalized and normalized cDNA microarray data. Basic methods for accessing/replacing, printing, and subsetting.

marrayInput
Functions for reading microarray data into R from different image analysis output files, and probe and target description files. Widgets are supplied to facilitate and automate data input and the creation of microarray specific R objects for storing these data.

marrayNorm
Functions for location and scale normalization procedures based on robust local regression.

marrayPlots
Functions for diagnostic plots for pre- and post-normalization cDNA microarray intensity data: boxplots, scatter-plots, color images.

marrayTools
Miscellaneous functions used in the functional genomics core facility in UCB and UCSF.

matchprobes
ools for sequence matching of probes on arrays.

multtest Multiple testing procedures for controlling the family-wise error rate (FWER) and the false discovery rate (FDR). Tests can be based on t- or F-statistics for one- and two-factor designs, and permutation procedures are available to estimate adjusted p-values.

ontoTools Graphs and sparse matrices for working with ontologies.

pamr Pam: prediction analysis for microarrays.

reposTools
Tools for dealing with file repositories and allow users to easily install, update, and distribute packages, vignettes, and other files.

rhdf5 Storage and retrieval of large datasets using the HDF5 library and file format.

siggenes Identifying differentially expressed genes and estimating the False Discovery Rate (FDR) with both the Significance Analysis of Microarrays (SAM) and the Empirical Bayes Analyses of Microarrays (EBAM).

splicegear A set of tools to work with alternative splicing.

tkWidgets Widgets in Tcl/Tk that provide functionality for Bioconductor packages.

vsn Calibration and variance stabilizing transformations for both Affymetrix and cDNA array data.

widgetTools
Tools for creating Tcl/Tk widgets, i.e., small-scale graphical user interfaces.

5.1.5 Other add-on packages

Jim Lindsey has written a collection of R packages for nonlinear regression and repeated measurements, consisting of **event** (event history procedures and models), **gnlm** (generalized nonlinear regression models), **growth** (multivariate normal and elliptically-contoured repeated measurements models), **repeated** (non-normal repeated measurements models), **rmutil** (utilities for nonlinear regression and repeated measurements), and **stable** (probability functions and generalized regression models for stable distributions). All analyses in the new edition of his book "Models for Repeated Measurements" (1999, Oxford University Press) were carried out using these packages. Jim has also started **dna**, a package with procedures for the analysis of DNA sequences. Jim's packages can be obtained from `http://www.luc.ac.be/~jlindsey/rcode.html`.

More code has been posted to the R-help mailing list, and can be obtained from the mailing list archive.

5.2 How can add-on packages be installed?

(Unix only.) The add-on packages on CRAN come as gzipped tar files named *pkg_version*`.tar.gz`, which may in fact be "bundles" containing more than one package. Provided that `tar` and `gzip` are available on your system, type

```
$ R CMD INSTALL /path/to/pkg_version.tar.gz
```

at the shell prompt to install to the library tree rooted at the first directory given in `R_LIBS` (see below) if this is set and non-null, and to the default library (the '`library`' subdirectory of '`R_HOME`') otherwise. (Versions of R prior to 1.3.0 installed to the default library by default.)

To install to another tree (e.g., your private one), use

```
$ R CMD INSTALL -l lib /path/to/pkg_version.tar.gz
```

where *lib* gives the path to the library tree to install to.

Even more conveniently, you can install and automatically update packages from within R if you have access to CRAN. See the help page for `CRAN.packages()` for more information.

You can use several library trees of add-on packages. The easiest way to tell R to use these is via the environment variable `R_LIBS` which should be a colon-separated list of directories at which R library trees are rooted. You do not have to specify the default tree in `R_LIBS`. E.g., to use a private tree in '`$HOME/lib/R`' and a public site-wide tree in '`/usr/local/lib/R-contrib`', put

```
R_LIBS="$HOME/lib/R:/usr/local/lib/R-contrib"; export R_LIBS
```

into your (Bourne) shell profile or even preferably, add the line

```
R_LIBS="$HOME/lib/R:/usr/local/lib/R-contrib"
```

your '`~/.Renviron`' file. (Note that no `export` statement is needed or allowed in this file; see the on-line help for `Startup` for more information.)

5.3 How can add-on packages be used?

To find out which additional packages are available on your system, type

```
library()
```

at the R prompt.

This produces something like

```
Packages in '/home/me/lib/R':

mystuff       My own R functions, nicely packaged but not documented

Packages in '/usr/local/lib/R/library':

KernSmooth    Functions for kernel smoothing for Wand & Jones (1995)
MASS          Main Library of Venables and Ripley's MASS
base          The R base package
boot          Bootstrap R (S-Plus) Functions (Canty)
class         Functions for classification
cluster       Functions for clustering (by Rousseeuw et al.)
ctest         Classical Tests
eda           Exploratory Data Analysis
foreign       Read data stored by Minitab, S, SAS, SPSS, Stata, ...
grid          The Grid Graphics Package
lattice       Lattice Graphics
lqs           Resistant Regression and Covariance Estimation
methods       Formal Methods and Classes
mle           Maximum likelihood estimation
mgcv          Multiple smoothing parameter estimation and GAMs by GCV
modreg        Modern Regression: Smoothing and Local Methods
mva           Classical Multivariate Analysis
nlme          Linear and nonlinear mixed effects models
nls           Nonlinear regression
nnet          Feed-forward neural networks and multinomial log-linear
              models
rpart         Recursive partitioning
spatial       functions for kriging and point pattern analysis
splines       Regression Spline Functions and Classes
stepfun       Step Functions, including Empirical Distributions
survival      Survival analysis, including penalised likelihood
tcltk         Tcl/Tk Interface
tools         Tools for Package Development and Administration
ts            Time series functions
```

You can "load" the installed package *pkg* by

```
library(pkg)
```

You can then find out which functions it provides by typing one of

```
library(help = pkg)
help(package = pkg)
```

You can unload the loaded package *pkg* by

```
detach("package:pkg")
```

5.4 How can add-on packages be removed?

Use

```
$ R CMD REMOVE pkg_1 ... pkg_n
```

to remove the packages *pkg_1*, ..., *pkg_n* from the library tree rooted at the first directory given in `R_LIBS` if this is set and non-null, and from the default library otherwise. (Versions of R prior to 1.3.0 removed from the default library by default.)

To remove from library *lib*, do

```
$ R CMD REMOVE -l lib pkg_1 ... pkg_n
```

5.5 How can I create an R package?

A package consists of a subdirectory containing the files '`DESCRIPTION`' and '`INDEX`', and the subdirectories '`R`', '`data`', '`demo`', '`exec`', '`inst`', '`man`', '`src`', and '`tests`' (some of which can be missing). Optionally the package can also contain script files '`configure`' and '`cleanup`' which are executed before and after installation.

See section "Creating R packages" in *Writing R Extensions*, for details. This manual is included in the R distribution, see Section 2.7 [What documentation exists for R?], page 6, and gives information on package structure, the configure and cleanup mechanisms, and on automated package checking and building.

R version 1.3.0 has added the function `package.skeleton()` which will set up directories, save data and code, and create skeleton help files for a set of R functions and datasets.

See Section 2.10 [What is CRAN?], page 9, for information on uploading a package to CRAN.

5.6 How can I contribute to R?

R is in active development and there is always a risk of bugs creeping in. Also, the developers do not have access to all possible machines capable of running R. So, simply using it and communicating problems is certainly of great value.

One place where functionality is still missing is the modeling software as described in "Statistical Models in S" (see Section 3.1 [What is S?], page 11); Generalized Additive Models (see Section 7.17 [Are GAMs implemented in R?], page 50) and some of the nonlinear modeling code are not there yet.

The R Developer Page acts as an intermediate repository for more or less finalized ideas and plans for the R statistical system. It contains (pointers to) TODO lists, RFCs, various other writeups, ideas lists, and CVS miscellanea.

Many (more) of the packages available at the Statlib S Repository might be worth porting to R.

If you are interested in working on any of these projects, please notify Kurt Hornik.

6 R and Emacs

6.1 Is there Emacs support for R?

There is an Emacs package called ESS ("Emacs Speaks Statistics") which provides a standard interface between statistical programs and statistical processes. It is intended to provide assistance for interactive statistical programming and data analysis. Languages supported include: S dialects (S 3/4, S-PLUS 3.x/4.x/5.x, and R), LispStat dialects (XLispStat, ViSta) and SAS. Stata and SPSS dialect (SPSS, PSPP) support is being examined for possible future implementation

ESS grew out of the need for bug fixes and extensions to S-mode 4.8 (which was a GNU Emacs interface to S/S-PLUS version 3 only). The current set of developers desired support for XEmacs, R, S4, and MS Windows. In addition, with new modes being developed for R, Stata, and SAS, it was felt that a unifying interface and framework for the user interface would benefit both the user and the developer, by helping both groups conform to standard Emacs usage. The end result is an increase in efficiency for statistical programming and data analysis, over the usual tools.

R support contains code for editing R source code (syntactic indentation and highlighting of source code, partial evaluations of code, loading and error-checking of code, and source code revision maintenance) and documentation (syntactic indentation and highlighting of source code, sending examples to running ESS process, and previewing), interacting with an inferior R process from within Emacs (command-line editing, searchable command history, command-line completion of R object and file names, quick access to object and search lists, transcript recording, and an interface to the help system), and transcript manipulation (recording and saving transcript files, manipulating and editing saved transcripts, and re-evaluating commands from transcript files).

The latest stable version of ESS are available via CRAN or the ESS web page. The HTML version of the documentation can be found at `http://stat.ethz.ch/ESS/`.

ESS comes with detailed installation instructions.

For help with ESS, send email to `ESS-help@stat.ethz.ch`.

Please send bug reports and suggestions on ESS to `ESS-bugs@stat.math.ethz.ch`. The easiest way to do this from is within Emacs by typing *`M-x ess-submit-bug-report`* or using the [ESS] or [iESS] pulldown menus.

6.2 Should I run R from within Emacs?

Yes, *definitely*. Inferior R mode provides a readline/history mechanism, object name completion, and syntax-based highlighting of the interaction buffer using Font Lock mode, as well as a very convenient interface to the R help system.

Of course, it also integrates nicely with the mechanisms for editing R source using Emacs. One can write code in one Emacs buffer and send whole or parts of it for execution to R; this is helpful for both data analysis and programming. One can also seamlessly integrate with a revision control system, in order to maintain a log of changes in your programs and data, as well as to allow for the retrieval of past versions of the code.

In addition, it allows you to keep a record of your session, which can also be used for error recovery through the use of the transcript mode.

To specify command line arguments for the inferior R process, use *C-u M-x R* for starting R.

6.3 Debugging R from within Emacs

To debug R "from within Emacs", there are several possibilities. To use the Emacs GUD (Grand Unified Debugger) library with the recommended debugger GDB, type *M-x gdb* and give the path to the R *binary* as argument. At the `gdb` prompt, set `R_HOME` and other environment variables as needed (using e.g. *set env R_HOME /path/to/R/*, but see also below), and start the binary with the desired arguments (e.g., *run --quiet*).

If you have ESS, you can do *C-u M-x R* ⟨RET⟩ *- d* ⟨SPC⟩ *g d b* ⟨RET⟩ to start an inferior R process with arguments '`-d gdb`'.

A third option is to start an inferior R process via ESS (*M-x R*) and then start GUD (*M-x gdb*) giving the R binary (using its full path name) as the program to debug. Use the program `ps` to find the process number of the currently running R process then use the `attach` command in gdb to attach it to that process. One advantage of this method is that you have separate `*R*` and `*gud-gdb*` windows. Within the `*R*` window you have all the ESS facilities, such as object-name completion, that we know and love.

When using GUD mode for debugging from within Emacs, you may find it most convenient to use the directory with your code in it as the current working directory and then make a symbolic link from that directory to the R binary. That way '`.gdbinit`' can stay in the directory with the code and be used to set up the environment and the search paths for the source, e.g. as follows:

```
set env R_HOME /opt/R
set env R_PAPERSIZE letter
set env R_PRINTCMD lpr
dir /opt/R/src/appl
dir /opt/R/src/main
dir /opt/R/src/nmath
dir /opt/R/src/unix
```

7 R Miscellanea

7.1 Why does R run out of memory?

Versions of R prior to 1.2.0 used a *static* memory model. At startup, R asked the operating system to reserve a fixed amount of memory for it. The size of this chunk could not be changed subsequently. Hence, it could happen that not enough memory was allocated, e.g., when trying to read large data sets into R. In such cases, it was necessary to restart R with more memory available, as controlled by the command line options '`--nsize`' and '`--vsize`'.

R version 1.2.0 introduces a new "generational" garbage collector, which will increase the memory available to R as needed. Hence, user intervention is no longer necessary for ensuring that enough memory is available.

The new garbage collector does not move objects in memory, meaning that it is possible for the free memory to become fragmented so that large objects cannot be allocated even when there is apparently enough memory for them.

7.2 Why does sourcing a correct file fail?

Versions of R prior to 1.2.1 may have had problems parsing files not ending in a newline. Earlier R versions had a similar problem when reading in data files. This should no longer happen.

7.3 How can I set components of a list to NULL?

You can use

```
x[i] <- list(NULL)
```

to set component `i` of the list `x` to `NULL`, similarly for named components. Do not set `x[i]` or `x[[i]]` to `NULL`, because this will remove the corresponding component from the list.

For dropping the row names of a matrix `x`, it may be easier to use `rownames(x) <- NULL`, similarly for column names.

7.4 How can I save my workspace?

`save.image()` saves the objects in the user's `.GlobalEnv` to the file '`.RData`' in the R startup directory. (This is also what happens after `q("yes")`.) Using `save.image(file)` one can save the image under a different name.

7.5 How can I clean up my workspace?

To remove all objects in the currently active environment (typically `.GlobalEnv`), you can do

```
rm(list = ls(all = TRUE))
```

(Without '`all = TRUE`', only the objects with names not starting with a '`.`' are removed.)

7.6 How can I get eval() and D() to work?

Strange things will happen if you use `eval(print(x), envir = e)` or `D(x^2, "x")`. The first one will either tell you that `"x"` is not found, or print the value of the wrong `x`. The other one will likely return zero if `x` exists, and an error otherwise.

This is because in both cases, the first argument is evaluated in the calling environment first. The result (which should be an object of mode `"expression"` or `"call"`) is then evaluated or differentiated. What you (most likely) really want is obtained by "quoting" the first argument upon surrounding it with `expression()`. For example,

```
R> D(expression(x^2), "x")
2 * x
```

Although this behavior may initially seem to be rather strange, is perfectly logical. The "intuitive" behavior could easily be implemented, but problems would arise whenever the expression is contained in a variable, passed as a parameter, or is the result of a function call. Consider for instance the semantics in cases like

```
D2 <- function(e, n) D(D(e, n), n)
```

or

```
g <- function(y) eval(substitute(y), sys.frame(sys.parent(n = 2)))
g(a * b)
```

See the help page for `deriv()` for more examples.

7.7 Why do my matrices lose dimensions?

When a matrix with a single row or column is created by a subscripting operation, e.g., `row <- mat[2, ]`, it is by default turned into a vector. In a similar way if an array with dimension, say, 2 x 3 x 1 x 4 is created by subscripting it will be coerced into a 2 x 3 x 4 array, losing the unnecessary dimension. After much discussion this has been determined to be a *feature*.

To prevent this happening, add the option '`drop = FALSE`' to the subscripting. For example,

```
rowmatrix <- mat[2, , drop = FALSE]  # creates a row matrix
colmatrix <- mat[, 2, drop = FALSE]  # creates a column matrix
a <- b[1, 1, 1, drop = FALSE]        # creates a 1 x 1 x 1 array
```

The '`drop = FALSE`' option should be used defensively when programming. For example, the statement

```
somerows <- mat[index, ]
```

will return a vector rather than a matrix if `index` happens to have length 1, causing errors later in the code. It should probably be rewritten as

```
somerows <- mat[index, , drop = FALSE]
```

7.8 How does autoloading work?

R has a special environment called `.AutoloadEnv`. Using *`autoload(name, pkg)`*, where *name* and *pkg* are strings giving the names of an object and the package containing it, stores some information in this environment. When R tries to evaluate *name*, it loads the corresponding package *pkg* and reevaluates *name* in the new package's environment.

Using this mechanism makes R behave as if the package was loaded, but does not occupy memory (yet).

See the help page for `autoload()` for a very nice example.

7.9 How should I set options?

The function `options()` allows setting and examining a variety of global "options" which affect the way in which R computes and displays its results. The variable `.Options` holds the current values of these options, but should never directly be assigned to unless you want to drive yourself crazy—simply pretend that it is a "read-only" variable.

For example, given

```
test1 <- function(x = pi, dig = 3) {
  oo <- options(digits = dig); on.exit(options(oo));
  cat(.Options$digits, x, "\n")
}
test2 <- function(x = pi, dig = 3) {
  .Options$digits <- dig
  cat(.Options$digits, x, "\n")
}
```

we obtain:

```
R> test1()
3 3.14
R> test2()
3 3.141593
```

What is really used is the *global* value of `.Options`, and using *`options(OPT = VAL)`* correctly updates it. Local copies of `.Options`, either in `.GlobalEnv` or in a function environment (frame), are just silently disregarded.

7.10 How do file names work in Windows?

As R uses C-style string handling, '\' is treated as an escape character, so that for example one can enter a newline as '`\n`'. When you really need a '\', you have to escape it with another '\'.

Thus, in filenames use something like `"c:\\data\\money.dat"`. You can also replace '\' by '/' (`"c:/data/money.dat"`).

7.11 Why does plotting give a color allocation error?

Sometimes plotting, e.g., when running `demo("image")`, results in "Error: color allocation error". This is an X problem, and only indirectly related to R. It occurs when applications started prior to R have used all the available colors. (How many colors are available depends on the X configuration; sometimes only 256 colors can be used.)

One application which is notorious for "eating" colors is Netscape. If the problem occurs when Netscape is running, try (re)starting it with either the '`-no-install`' (to use the default colormap) or the '`-install`' (to install a private colormap) option.

You could also set the `colortype` of `X11()` to `"pseudo.cube"` rather than the default `"pseudo"`. See the help page for `X11()` for more information.

7.12 How do I convert factors to numeric?

It may happen that when reading numeric data into R (usually, when reading in a file), they come in as factors. If `f` is such a factor object, you can use

```
as.numeric(as.character(f))
```

to get the numbers back. More efficient, but harder to remember, is

```
as.numeric(levels(f))[as.integer(f)]
```

In any case, do not call `as.numeric()` or their likes directly for the task at hand (as `as.numeric()` or `unclass()` give the internal codes).

7.13 Are Trellis displays implemented in R?

The recommended package **lattice** (which is based on another recommended package, **grid**) provides graphical functionality that is compatible with most Trellis commands.

You could also look at `coplot()` and `dotchart()` which might do at least some of what you want. Note also that the R version of `pairs()` is fairly general and provides most of the functionality of `splom()`, and that R's default plot method has an argument `asp` allowing to specify (and fix against device resizing) the aspect ratio of the plot.

(Because the word "Trellis" has been claimed as a trademark we do not use it in R. The name "lattice" has been chosen for the R equivalent.)

7.14 What are the enclosing and parent environments?

Inside a function you may want to access variables in two additional environments: the one that the function was defined in ("enclosing"), and the one it was invoked in ("parent").

If you create a function at the command line or load it in a package its enclosing environment is the global workspace. If you define a function `f()` inside another function `g()` its enclosing environment is the environment inside `g()`. The enclosing environment for a

function is fixed when the function is created. You can find out the enclosing environment for a function `f()` using `environment(f)`.

The "parent" environment, on the other hand, is defined when you invoke a function. If you invoke `lm()` at the command line its parent environment is the global workspace, if you invoke it inside a function `f()` then its parent environment is the environment inside `f()`. You can find out the parent environment for an invocation of a function by using `parent.frame()` or `sys.frame(sys.parent())`.

So for most user-visible functions the enclosing environment will be the global workspace, since that is where most functions are defined. The parent environment will be wherever the function happens to be called from. If a function `f()` is defined inside another function `g()` it will probably be used inside `g()` as well, so its parent environment and enclosing environment will probably be the same.

Parent environments are important because things like model formulas need to be evaluated in the environment the function was called from, since that's where all the variables will be available. This relies on the parent environment being potentially different with each invocation.

Enclosing environments are important because a function can use variables in the enclosing environment to share information with other functions or with other invocations of itself (see the section on lexical scoping). This relies on the enclosing environment being the same each time the function is invoked.

Scoping *is* hard. Looking at examples helps. It is particularly instructive to look at examples that work differently in R and S and try to see why they differ. One way to describe the scoping differences between R and S is to say that in S the enclosing environment is *always* the global workspace, but in R the enclosing environment is wherever the function was created.

7.15 How can I substitute into a plot label?

Often, it is desired to use the value of an R object in a plot label, e.g., a title. This is easily accomplished using `paste()` if the label is a simple character string, but not always obvious in case the label is an expression (for refined mathematical annotation). In such a case, either use `parse()` on your pasted character string or use `substitute()` on an expression. For example, if `ahat` is an estimator of your parameter a of interest, use

```
title(substitute(hat(a) == ahat, list(ahat = ahat)))
```

(note that it is '`==`' and not '`=`'). There are more worked examples in the mailing list achives.

7.16 What are valid names?

When creating data frames using `data.frame()` or `read.table()`, R by default ensures that the variable names are syntactically valid. (The argument '`check.names`' to these functions controls whether variable names are checked and adjusted by `make.names()` if needed.)

To understand what names are "valid", one needs to take into account that the term "name" is used in several different (but related) ways in the language:

1. A *syntactic name* is a string the parser interprets as this type of expression. It consists of letters, numbers, and the dot character and starts with a letter or the dot.
2. An *object name* is a string associated with an object that is assigned in an expression either by having the object name on the left of an assignment operation or as an argument to the `assign()` function. It is usually a syntactic name as well, but can be any non-empty string if it is quoted (and it is always quoted in the call to `assign()`).
3. An *argument name* is what appears to the left of the equals sign when supplying an argument in a function call (for example, `f(trim=.5)`). Argument names are also usually syntactic names, but again can be anything if they are quoted.
4. An *element name* is a string that identifies a piece of an object (a component of a list, for example.) When it is used on the right of the '`$`' operator, it must be a syntactic name, or quoted. Otherwise, element names can be any strings. (When an object is used as a database, as in a call to `eval()` or `attach()`, the element names become object names.)
5. Finally, a *file name* is a string identifying a file in the operating system for reading, writing, etc. It really has nothing much to do with names in the language, but it is traditional to call these strings file "names".

7.17 Are GAMs implemented in R?

There is a `gam()` function for Generalized Additive Models in package **mgcv**, but it is not an exact clone of what is described in the White Book (no `lo()` for example). Package **gss** can fit spline-based GAMs too. And if you can accept regression splines you can use `glm()`. For gaussian GAMs you can use `bruto()` from package **mda**.

7.18 Why is the output not printed when I source() a file?

Most R commands do not generate any output. The command

```
1+1
```

computes the value 2 and returns it; the command

```
summary(glm(y~x+z, family=binomial))
```

fits a logistic regression model, computes some summary information and returns an object of class `"summary.glm"` (see Section 8.1 [How should I write summary methods?], page 54).

If you type '`1+1`' or '`summary(glm(y~x+z, family=binomial))`' at the command line the returned value is automatically printed (unless it is `invisible()`), but in other circumstances, such as in a `source()`d file or inside a function it isn't printed unless you specifically print it.

To print the value use

```
print(1+1)
```

or

```
print(summary(glm(y~x+z, family=binomial)))
```

instead, or use `source(`*`file`*`, echo=TRUE)`.

7.19 Why does outer() behave strangely with my function?

As the help for `outer()` indicates, it does not work on arbitrary functions the way the `apply()` family does. It requires functions that are vectorized to work elementwise on arrays. As you can see by looking at the code, `outer(x, y, FUN)` creates two large vectors containing every possible combination of elements of `x` and `y` and then passes this to `FUN` all at once. Your function probably cannot handle two large vectors as parameters.

If you have a function that cannot handle two vectors but can handle two scalars, then you can still use `outer()` but you will need to wrap your function up first, to simulate vectorized behavior. Suppose your function is

```
foo <- function(x, y, happy) {
  stopifnot(length(x) == 1, length(y) == 1) # scalars only!
  (x + y) * happy
}
```

If you define the general function

```
wrapper <- function(x, y, my.fun, ...) {
  sapply(seq(along = x), FUN = function(i) my.fun(x[i], y[i], ...))
}
```

then you can use `outer()` by writing, e.g.,

```
outer(1:4, 1:2, FUN = wrapper, my.fun = foo, happy = 10)
```

7.20 Why does the output from anova() depend on the order of factors in the model?

In a model such as `~A+B+A:B`, R will report the difference in sums of squares between the models `~1`, `~A`, `~A+B` and `~A+B+A:B`. If the model were `~B+A+A:B`, R would report differences between `~1`, `~B`, `~A+B`, and `~A+B+A:B` . In the first case the sum of squares for `A` is comparing `~1` and `~A`, in the second case it is comparing `~B` and `~B+A`. In a non-orthogonal design (i.e., most unbalanced designs) these comparisons are (conceptually and numerically) different.

Some packages report instead the sums of squares based on comparing the full model to the models with each factor removed one at a time (the famous 'Type III sums of squares' from SAS, for example). These do not depend on the order of factors in the model. The question of which set of sums of squares is the Right Thing provokes low-level holy wars on R-help from time to time.

There is no need to be agitated about the particular sums of squares that R reports. You can compute your favorite sums of squares quite easily. Any two models can be compared with `anova(`*`model1, model2`*`)`, and `drop1(`*`model1`*`)` will show the sums of squares resulting from dropping single terms.

7.21 How do I produce PNG graphics in batch mode?

Under Unix, the `png()` device uses the X11 driver, which is a problem in batch mode or for remote operation. If you have Ghostscript you can use `bitmap()`, which produces a PostScript file then converts it to any bitmap format supported by ghostscript. On some

installations this produces ugly output, on others it is perfectly satisfactory. In theory one could also use Xvfb, which provides an X server with no display.

7.22 How can I get command line editing to work?

The Unix command-line interface to R can only provide the inbuilt command line editor which allows recall, editing and re-submission of prior commands provided that the GNU readline library is available at the time R is configured for compilation. Note that the 'development' version of readline including the appropriate headers is needed: users of Linux binary distributions will need to install packages such as `libreadline-dev` (Debian) or `readline-devel` (Red Hat).

7.23 How can I turn a string into a variable?

If you have

```
varname <- c("a", "b", "d")
```

you can do

```
get(varname[1]) + 2
```

for

```
a + 2
```

or

```
assign(varname[1], 2 + 2)
```

for

```
a <- 2 + 2
```

or

```
eval(substitute(lm(y ~ x + variable),
                list(variable = as.name(varname[1]))
```

for

```
lm(y ~ x + a)
```

At least in the first two cases it is often easier to just use a list, and then you can easily index it by name

```
vars <- list(a = 1:10, b = rnorm(100), d = LETTERS)
vars[["a"]]
```

without any of this messing about.

7.24 Why do lattice/trellis graphics not work?

The most likely reason is that you forgot to tell R to display the graph. Lattice functions such as `xyplot()` create a graph object, but do not display it (the same is true of Trellis graphics in S-PLUS). The `print()` method for the graph object produces the actual display. When you use these functions interactively at the command line, the result is automatically printed, but in `source()` or inside your own functions you will need an explicit `print()` statement.

7.25 How can I sort the rows of a data frame?

To sort the rows within a data frame, with respect to the values in one or more of the columns, simply use `order()`.

8 R Programming

8.1 How should I write summary methods?

Suppose you want to provide a summary method for class `"foo"`. Then `summary.foo()` should not print anything, but return an object of class `"summary.foo"`, *and* you should write a method `print.summary.foo()` which nicely prints the summary information and invisibly returns its object. This approach is preferred over having `summary.foo()` print summary information and return something useful, as sometimes you need to grab something computed by `summary()` inside a function or similar. In such cases you don't want anything printed.

8.2 How can I debug dynamically loaded code?

Roughly speaking, you need to start R inside the debugger, load the code, send an interrupt, and then set the required breakpoints.

See section "Finding entry points in dynamically loaded code" in *Writing R Extensions*. This manual is included in the R distribution, see Section 2.7 [What documentation exists for R?], page 6.

8.3 How can I inspect R objects when debugging?

The most convenient way is to call `R_PV` from the symbolic debugger.

See section "Inspecting R objects when debugging" in *Writing R Extensions*.

8.4 How can I change compilation flags?

Suppose you have C code file for dynloading into R, but you want to use `R CMD SHLIB` with compilation flags other than the default ones (which were determined when R was built). You could change the file '`R_HOME/etc/Makeconf`' to reflect your preferences. If you are a Bourne shell user, you can also pass the desired flags to Make (which is used for controlling compilation) via the Make variable `MAKEFLAGS`, as in

```
MAKEFLAGS="CFLAGS=-O3" R CMD SHLIB *.c
```

9 R Bugs

9.1 What is a bug?

If R executes an illegal instruction, or dies with an operating system error message that indicates a problem in the program (as opposed to something like "disk full"), then it is certainly a bug. If you call `.C()`, `.Fortran()`, `.External()` or `.Call()` (or `.Internal()`) yourself (or in a function you wrote), you can always crash R by using wrong argument types (modes). This is not a bug.

Taking forever to complete a command can be a bug, but you must make certain that it was really R's fault. Some commands simply take a long time. If the input was such that you *know* it should have been processed quickly, report a bug. If you don't know whether the command should take a long time, find out by looking in the manual or by asking for assistance.

If a command you are familiar with causes an R error message in a case where its usual definition ought to be reasonable, it is probably a bug. If a command does the wrong thing, that is a bug. But be sure you know for certain what it ought to have done. If you aren't familiar with the command, or don't know for certain how the command is supposed to work, then it might actually be working right. Rather than jumping to conclusions, show the problem to someone who knows for certain.

Finally, a command's intended definition may not be best for statistical analysis. This is a very important sort of problem, but it is also a matter of judgment. Also, it is easy to come to such a conclusion out of ignorance of some of the existing features. It is probably best not to complain about such a problem until you have checked the documentation in the usual ways, feel confident that you understand it, and know for certain that what you want is not available. If you are not sure what the command is supposed to do after a careful reading of the manual this indicates a bug in the manual. The manual's job is to make everything clear. It is just as important to report documentation bugs as program bugs. However, we know that the introductory documentation is seriously inadequate, so you don't need to report this.

If the online argument list of a function disagrees with the manual, one of them must be wrong, so report the bug.

9.2 How to report a bug

When you decide that there is a bug, it is important to report it and to report it in a way which is useful. What is most useful is an exact description of what commands you type, starting with the shell command to run R, until the problem happens. Always include the version of R, machine, and operating system that you are using; type `version` in R to print this.

The most important principle in reporting a bug is to report *facts*, not hypotheses or categorizations. It is always easier to report the facts, but people seem to prefer to strain to posit explanations and report them instead. If the explanations are based on guesses about how R is implemented, they will be useless; others will have to try to figure out what

the facts must have been to lead to such speculations. Sometimes this is impossible. But in any case, it is unnecessary work for the ones trying to fix the problem.

For example, suppose that on a data set which you know to be quite large the command

```
R> data.frame(x, y, z, monday, tuesday)
```

never returns. Do not report that `data.frame()` fails for large data sets. Perhaps it fails when a variable name is a day of the week. If this is so then when others got your report they would try out the `data.frame()` command on a large data set, probably with no day of the week variable name, and not see any problem. There is no way in the world that others could guess that they should try a day of the week variable name.

Or perhaps the command fails because the last command you used was a method for `"["()` that had a bug causing R's internal data structures to be corrupted and making the `data.frame()` command fail from then on. This is why others need to know what other commands you have typed (or read from your startup file).

It is very useful to try and find simple examples that produce apparently the same bug, and somewhat useful to find simple examples that might be expected to produce the bug but actually do not. If you want to debug the problem and find exactly what caused it, that is wonderful. You should still report the facts as well as any explanations or solutions. Please include an example that reproduces the problem, preferably the simplest one you have found.

Invoking R with the '`--vanilla`' option may help in isolating a bug. This ensures that the site profile and saved data files are not read.

On Unix systems a bug report can be generated using the function `bug.report()`. This automatically includes the version information and sends the bug to the correct address. Alternatively the bug report can be emailed to `R-bugs@R-project.org` or submitted to the Web page at `http://bugs.R-project.org/`.

Bug reports on contributed packages should perhaps be sent to the package maintainer rather than to R-bugs.

10 Acknowledgments

Of course, many many thanks to Robert and Ross for the R system, and to the package writers and porters for adding to it.

Special thanks go to Doug Bates, Peter Dalgaard, Paul Gilbert, Stefano Iacus, Fritz Leisch, Jim Lindsey, Thomas Lumley, Martin Maechler, Brian D. Ripley, Anthony Rossini, and Andreas Weingessel for their comments which helped me improve this FAQ.

More to some soon . . .