A

Glossary of R jargon

Below is a selection of common R terms defined first using Stata jargon (or plain English when possible) and then more formally using R jargon. Some definitions in Stata jargon are quite loose given the fact that they have no direct analog of some R terms. Definitions in R terms are often quoted (with permission) or paraphrased from *S Poetry* by Patrick Burns [3].

Apply
: The process of having a command work on variables or observations. Determines whether a procedure will act as a typical command or as a function instead. Also the name of a function that controls that process. More formally, the process of targeting a function on rows or columns. Also a function that does that.

Argument
: The options that control what the commands do and the arguments that control what functions do. Confusing because in R, functions do what both commands and functions do in Stata. More formally, input(s) to a function that control it. Includes data to analyze.

Array
: A matrix with more than two dimensions. All variables must be only one type (e.g., all numeric or all character). More formally, a vector with a dim attribute. The dim controls the number and size of dimensions.

Assignment function
: Assigns values like the equal sign in Stata. The two-key sequence, "`<-`", that places data or results of procedures or transformations into a variable or data set. More formally, the two-key sequence, "`<-`", that gives names to objects.

Atomic object
: A variable whose values are all of one type, such as all numeric or all character. More formally, an object whose components are all of one mode. Modes allowed are numeric, character, logical, or complex.

R.A. Muenchen, J.M. Hilbe, *R for Stata Users*, Statistics and Computing, DOI 10.1007/978-1-4419-1318-0,

Attach
: The process of adding a data set or add-on module to your path. Attaching a data set appears to copy the variables into an area that lets you use them by a simple component name like "gender" rather than by using the $ format name like "mydata$gender." Done using the `attach` function. More formally, the process of adding a database to your search list. Also a function that does this.

Attributes
: Traits of a data set like its variable names and labels. More formally, traits of objects such as names, class, or dim.

Class
: An attribute of a variable or data set that a command used to change its options automatically. More formally, the class attribute of an object determines which method of a generic function is used when the object is an argument in the function call.

Component
: Like one data set stored in a zipped set of data sets. More formally, an item in a list. The length of a list is the number of components it has.

CRAN
: The Comprehensive R Archive Network at `http://cran.r-project.org/`. An Internet archive like the Statistical Software Components (SSC) Archive. Consists of a set of sites around the world called mirrors that provide R and its add-on packages for you to download and install.

Data frame
: A data set. More formally, a set of vectors bound together in a list. They can be different modes or classes (e.g., numeric and character), but they must have equal length.

Database
: One data set or a set of them, or an add-on module. More formally, an item on the search list or something that might be. Can be an R data file or a package.

Dim
: A variable whose values are the number of rows and columns in a data set. It is stored in the data set itself. Also, a procedure that prints or sets these values. More formally, the attribute that describes the *dim*ensions of an array. Also, the function that retrieves or changes that attribute.

Element
: A specific value for a variable. More formally, an item in a vector.

Extractor function
: A postestimation command. More formally, a function that has methods that apply to modeling objects.

Factor
: A categorical variable and its value labels. Value labels may be nothing more than "1," "2,"..., if not assigned explicitly. More formally, the

type of object that represents a categorical variable. It stores its labels in its levels attribute.

Function

A command and/or a function. When you apply it down through cases, it is just like a Stata command. However, you can also apply it across rows like a Stata function. More formally, an R program that is stored as an object.

Generic function

A command or function that has different default options or arguments set depending on the type of data you give it. More formally, a function whose behavior is determined by the class of one or more of its arguments. The class of the relevant argument(s) determines which method the generic function will use.

Index

The order number of a variable in a data set or the subscript of a value in a variable. In our practice data set gender is the second variable, so its index is 2. Gender is `mydata[ ,2]`. The first index selects rows, the second selects columns. If empty, it refers to all rows/columns. More formally, the number of the component in a list or data frame, or of an element in a vector.

Install

You install packages just like ado files, just once per version of R. However, you must load it from the library every time you start R. More formally, adding a package into your library.

Label

A procedure that creates variable labels. Also, a parameter that sets value labels using the `factor` or `ordered` commands. More formally, a function from the `Hmisc` package that creates variable labels. Also an argument that sets factor labels using the `factor` or `ordered` functions.

Length

The number of observations/cases in a variable, including missing values, or the number of variables in a data set. More formally, a measure of objects. For vectors, it is the number of its elements (including NAs). For lists or data frames, it is the number of its components.

Levels

The values that a categorical variable can have. Actually stored as a part of the variable itself in what appears to be a very short character variable (even when the values themselves are numbers). More formally, an attribute to a factor object that is a character vector of the values the factor can have. Also an argument to the `factor` and `ordered` functions that can set the levels.

Library

Where a given version of R stores its base packages and the add-on modules you have installed. Also a procedure that loads a package from the library into working memory. You must do that in every R

session before using a package. More formally, a directory containing R packages that is set up so that the library function can attach it. Also a function that attaches a package from the library onto your search list.

List

Like a zipped collection of data sets that you can analyze easily without unzipping. More formally, a set of objects of any class. Can contain vectors, data frames, matrices and even other lists.

Load

Bringing a data set (or collection of data sets) from disk to memory. You must do this before you can use data in R. Also the command that performs that task, like the Stata `use` command. More formally, bringing an R data file into your workspace. Also the function that performs that task.

Matrix

A data set that must contain only one type of variable, e.g. all numeric or character. More formally, a two-dimensional array; that is, a vector with a dim attribute of length 2.

Method

The analyses and/or graphs that a procedure will perform by default, that is different for different types of variables. The default settings for some commands depend on the scale of the variables you provide. E.g. summary(temperature) provides mean temperature, summary(gender) counts males & females. More formally, a function that provides the calculation of a generic function for a specific class of object.

Mode

A variable's type such as numeric or character. More formally, a fundamental property of an object. Can be numeric, character, logical or complex.

Modeling function

A command that performs estimation. More formally, a function that tests association or group differences and usually accepts a formula (e.g. `y~x`) and a `data=` argument.

Modeling objects

A model created by a modeling function.

NA

A missing value. Stands for *N*ot *A*vailable. See also NaN.

Names

Variable names. They are stored in a character variable that is a part of a data set or variable. Since R can use an index number instead, names are optional. Also a procedure that extracts or changes variable names. More formally, an attribute of many objects that labels the elements or components of the object. Also the function that retrieves or sets this attribute.

NaN
: A missing value. Stands for *N*ot a *N*umber. Something that is undefined mathematically such as zero divided by zero.

NULL
: An object you can use to drop variables or values. E.g. mydata$x ¡- NULL drops the variable x from the data set mydata. More formally, NULL has a zero length and no particular mode. Assigning it to an object deletes it.

Numeric
: A variable that contains only numbers. More formally, the atomic mode that represents real numbers. This contains storage modes double, single and integer.

Object
: A data set, a variable or even the equivalent of a Stata command). More formally, almost everything in R. If it has a mode, it is an object. Includes data frames, vectors, matrices, arrays, lists and functions.

Object Oriented Programming
: A style of software in which the output of a procedure depends on the type of data you provide it. R has an object orientation, Stata added it in version 11.

Option
: A statement that sets general parameters, such as the width of each line of output. More formally, settings that control some aspect of your R session, such as the width of each line of output. Also a function that queries or changes the settings.

Package
: A set of ado-files, and related files, such as help, for each bundled together. Like the packages at the SSC. May come with R or be written by its users. More formally, a collection of functions and, optionally, data objects.

R
: A language and environment for statistical computing and graphics. An implementation of the S language.

R-PLUS
: A commercial version of R. It includes a graphical user interface, context-sensitive editor and other features.

Replacement
: A way to replace values. More formally, when you use subscripts on the left side of an assignment to change the values in an object. E.g. setting 9 to missing: `x[x==9] <- NA`

S
: The language from which R evolved. R can run many S programs, but S cannot use R packages.

S3, S4
: Used in the R help files to refer to different versions of S. The differences between them are of importance mainly to advanced programmers.

Script
: The equivalent of a do file. An R program.

Search list
: Somewhat like an operating system search path for R objects. More formally, the collection of databases that R will search, in order, for objects.

S-PLUS
: The commercial version of S. Mostly compatible with R but will not run R packages. It includes graphical user interface and can analyze "big data" that is larger than your computer's main memory.

Subscript
: Choosing variables or values by the order in which they appear or by their name. More formally, the extraction or replacement of an object using its index or name in square [brackets].

Vector
: A variable. It can exist on its own in memory or it can be part of a data set. More formally, a set of values that have the same mode, i.e. an atomic object.

Workspace
: A temporary work area in which all R computation happens. Data that exists there will vanish if not saved to your hard drive before quitting R. More formally, the area of your computer's main memory where R does all its work. Data must be loaded into it from files, and packages must be loaded into it from the library, before you can use either.

B

Comparison of Stata commands and R functions

With over 3,000 add-on packages, many containing multiple procedures, R can do almost everything that Stata can do and quite a bit more. People are releasing new packages at a rapid pace and R can give you the latest count with the following program.

The first function sets the repositories for R to search. A dialog box will prompt you so you can select them all. Next, the `available.packages` function searches the Internet repositories for the packages that are currently available, and stores their names in myPackageNames. Finally, the `unique` function counts the number of unique package names.

```
> setRepositories()
      (select all of the repositories it offers)
> myPackageNames <- available.packages()
> length(unique( rownames(myPackageNames) ))

[1] 3175
```

So at the time of publication, there were 3,175 add-on packages.

The table below focuses only on a small but important subset of areas. Much more detailed information about R packages is available organized in Task Views at `http://cran.r-project.org/web/views/index.html`. Another site to search by task is at `http://biostat.mc.vanderbilt.edu/s/finder/finder.html`. Detailed information about most R packages is available at http://www.r-project.org/, choose CRAN, then choose a mirror, then choose Packages.

Table B.1. Comparison of Stata commands and functions to R functions.

Stata command or function	R function (or package name)
* comment	# comment
append	rbind.fill (reshape); rbind
anova; oneway	aov
bitest	binom.test
ci	confint
clear	rm(list=ls())
correlate	cor, corr.test, corr (Hmisc)
describe	attributes; str; ls.str; contents (Hmisc)
drop	mydata["varname"]<-NULL
edit	fix; edit
findit	help.search
help	help.start
help <topic>	help(<topic>) or ?<topic>
include	source
glm	glm
graph	plot; qplot (Hmisc); ggplot (Hmisc)
infile; infix	read.table; read.csv; read.FWF
keep	Select the variables using any technique.
kwallis	kruskal.test
label values	factor; ordered
label variables	label (Hmisc)
list	print; head; tail;
logistic	glm(...family=binomial)
mean	mean
merge	merge
nbreg	glm.nb()
ologit	polr
outsheet	write.table
poisson	glm(...family=poisson)
predict	predict
qnorm	qqnorm
recode	recode (car)
rename	rename (reshape); names
regress	lm
reshape	reshape; melt/cast (reshape)
save	save; save.image
set	options
search	help.search
ssc	install.packages
signrank	wilcox.test
sort	order
sum	sum
summarize	summary
tab <x>, gen()	factor, ordered
table	CrossTable (gmodels); table
tabulate	table
tabstat	tapply; by; aggregate
ttest	t.test
type	file.show
update	update.packages
use	load
xtgee	gee

C

Automating Your R Setup

Stata has the *profile.do* file lets you automatically set options. R has a similar file called .Rprofile. This file is stored in your initial working directory, which you can locate with the `getwd()` function.

We will look at some useful things to automate in an .Rprofile.

C.1 Setting Options

In your .Rprofile, you can set options just as you would in R. I usually set my console width to 64 so the output fits training examples better. I also ask for five significant digits and tell it to mark significant results with stars. The latter is the default, but since many people prefer to turn that feature off, I included it. You would turn them off with a setting of FALSE.

```
options(width=64, digits=5, show.signif.stars=TRUE)
```

Enter `help(options)` for a comprehensive list of parameters that you can set using the `options` function.

Setting the random number seed is a good idea if you want to generate numbers that are random but repeatable. That is handy for training examples in which you would like every student to see the same result. Here I set it to the number 1234.

```
set.seed(1234)
```

The `setwd` function sets the working directory, the place that all of your files will go if you do not specify a path.

```
setwd("/myRfolder")
```

Since I included the "/" in the working directory path, it will go to the root level of my hard drive. That works in most operating systems. Note that it must be a forward slash, even in Windows, which usually uses backward slashes in filenames. If you leave the slash off completely, it will set it to be a folder within your normal working directory.

C.2 Creating Objects

We also like to define the set of packages that we install whenever we upgrade to a new version of R. With these stored in myPackages, I can install them all with a single function call. For details, see Chapter 2, "Installing and Updating R." This is the list of some of the packages used in this book.

```
myPackages <- c("car","hexbin","ggplot2",
  "gmodels","gplots", "Hmisc",
  "reshape","Rcmdr","prettyR")
```

C.3 Loading Packages

You can have R load load your favorite packages automatically too. This is particularly helpful when setting up a computer to run R with a graphical user interface like R Commander. Loading packages at startup does have some disadvantages though. It slows down your startup time, takes up memory in your workspace, and can create conflicts when different packages have functions with the same name. Therefore, you do not want to load too many this way.

Loading packages at startup requires the use of the `local` function. The `getOption` function gets the names of the original packages to load and stores them in a character vector I named myOriginal. I then created a second character vector, myAutoLoads, containing the names of the packages I want to add to the list. I then combined them into one character vector, myBoth. Finally, I used the `options` function to change the default packages to the combined list of both the original list and my chosen packages:

```
local({
   myOriginal <- getOption("defaultPackages")

   # edit next line to be your list of favorites.
   myAutoLoads <- c("Hmisc","ggplot2")

   myBoth <- c(myOriginal,myAutoLoads)

   options(defaultPackages = myBoth)
 })
```

C.4 Running Functions

If you want R to run any functions automatically, you create your own single functions that do the required steps. To have R run a function before all

others, name it ".First." To have it run the function after all others, name it ".Last." Notice that utility functions require a prefix of "utils::" or R will not find them while it is starting up. The timestamp function is one of those. It returns the time and date. The cat function prints messages. Its name comes from the UNIX command "cat". It is short for *catenate* (a synonym for concatenate). In essence, we will use it to concatenate the timestamp to your console output.

```
.First <- function()
  {
    cat("\n              Welcome to R!\n")
    utils::timestamp()
    cat("\n")
  }
```

You can also have R run any functions before exiting the package. I have it turn off my graphics device drivers with the graphics.off function to ensure that no files are left open.

I like to have it save my command history in case I later decide I should have saved some of the commands to a script file. Below I print a farewell message and then save the history to a file named myLatest.Rhistory.

```
.Last <- function()
  {
    graphics.off() #turns off graphics devices just in case.
    cat("\n\n  myCumulative.Rhistory has been saved." )
    cat("\n\n  Goodbye!\n\n")
    utils::savehistory(file="myCumulative.Rhistory")
  }
```

WARNING: Since the .First and .Last functions begin with a period, they are invisible to the ls function by default. The function call

```
ls(all.names=TRUE)
```

will show them to you. Since they are functions, if you save a workspace that contains them, they will continue to operate whenever you load that workspace, even if you delete the .Rprofile! This can make it *very* difficult to debug a problem until you realize what is happening. As usual, you can display them by typing their names and run them by adding empty parentheses to them:

```
.First()
```

If you need to delete them from the workspace, rm will do it with no added arguments:

```
rm(.First,.Last)
```

C.5 Example .Rprofile

The following is the .Rprofile with all of the above function calls combined. You do not have to type this in; it is included in the book's programs and data files at `http://r4stats.com`.

```
#     Startup Settings
# Place any R commands below.

options(width=64, digits=5, show.signif.stars=TRUE)
set.seed(1234)
setwd("/myRfolder")
myPackages <- c("car","hexbin","ggplot2",
  "gmodels","gplots", "Hmisc",
  "reshape","ggplot2","Rcmdr")
utils::loadhistory(file = "myCumulative.Rhistory")

# Load packages automatically below.

 local({
   myOriginal <- getOption("defaultPackages")

   # Edit next line to include your favorites.
   myAutoLoads <- c("Hmisc","ggplot2")
   myBoth <- c(myOriginal,myAutoLoads)
   options(defaultPackages = myBoth)
 })

# Things put here are done first.
.First <- function()
  {
    cat("\n              Welcome to R!\n")
    utils::timestamp()
    cat("\n")
  }

# Things put here are done last.
.Last <- function()
  {
    graphics.off()
    cat("\n\n  myCumulative.Rhistory has been saved." )
    cat("\n\n  Goodbye!\n\n")
    utils::savehistory(file="myCumulative.Rhistory")
  }
```

D

Example Simulation

The following examples are fully working program files. When run, they each create the same synthetic logistic regression data set consisting of 50,000 observations and a response or dependent variable with two normally distributed continuous predictors, x1 and x2. The values assigned to the predictors and intercept are

x1 = 0.75 x2 = −1.25 intercept or constant = 3

The binary response, or dependent variable, is created using a binomial random number generator, based on the linear predictor, xb, which is created from the randomly generated data.

Once the data has been created, it is estimated using the GLM functions of the two software applications. For details, see Joseph Hilbe's article, Creation of Synthetic Discrete Response Regression Models [22].

D.1 Stata Example Simulation

```
* Filename: GenerateLogit.do

clear
set obs 50000
set seed 13579
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
gen xb = 2 + 0.75*x1 - 1.25*x2
gen exb = 1/(1+exp(-xb))
gen by = rbinomial(1, exb)
glm by x1 x2, nolog fam(bin 1)
```

D.2 R Example Simulation

```
# Filename: GenerateLogit.R

library(MASS)
x1 <- runif(50000)
x2 <- runif(50000)
xb <- 2 + .75*x1 - 1.25*x2
exb <- 1/(1+exp(-xb))
by <- rbinom(50000, size = 1, prob =exp)

lry <- glm(by ~ x1 + x2, family=binomial(link="logit"))
summary(lry)
```

References

[1] Carlos Alzola and Jr. Frank E. Harrell. *An Introduction to S and the Hmisc and Design Libraries*.
Available from http://biostat.mc.vanderbilt.edu/RS/sintro.pdf, 2006.

[2] Richard A. Becker, Allan R. Wilks, Ray Brownrigg, and Thomas P Minka. *maps: Draw Geographical Maps*.
Available from http://cran.r-project.org, 2009. R package version 2.1-0.

[3] Burns. *S poetry*.
Available from http://www.burns-stat.com/pages/spoetry.html, 1998.

[4] Dan Carr, ported by Nicholas Lewin-Koh, and Martin Maechler. *hexbin: Hexagonal Binning Routines*.
Available from http://cran.r-project.org, 2008.

[5] John M. Chambers. *Software for Data Analysis: Programming with R*. Springer, 2008.

[6] W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[7] R core members, Saikat DebRoy, Roger Bivand, et al. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...*
Available from http://cran.r-project.org, 2009. R package version 0.8-33.

[8] David B. Dahl. *xtable: Export tables to LaTeX or HTML*.
Available from http://cran.r-project.org, 2009. R package version 1.5-5.

[9] Peter Dalgaard. *Introductory Statistics with R (Statistics and Computing)*. Springer, 2008.

[10] Martin Elff. *memisc: Tools for Management of Survey Data, Graphics, Programming, Statistics, and Simulation*.
Available from http://cran.r-project.org, 2009. R package version 0.95-22.

[11] The R Foundation for Statistical Computing. R: Regulatory compliance and validation issues a guidance document for the use of r in regulated clinical trial environments, 2008. Available from http://www.r-project.org/doc/R-FDA.pdf.

[12] John Fox. *car: Companion to Applied Regression*.
Available from http://cran.r-project.org, 2009. R package version 1.2-12.

[13] John Fox and with contributions from many others. *Rcmdr: R Commander*. Available from http://CRAN.R-project.org, 2009. R package version 1.4-7.

[14] Jr. Frank E. Harrell and with contributions from many other users. *Hmisc: Harrell Miscellaneous*.
Available from http://cran.r-project.org, 2008. R package version 3.5-2.

[15] Michael Friendly. *Visualizing Categorical Data: Data, Stories, and Pictures*. SAS Publishing, 2000.

[16] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[17] Kenneth W. Haemer. Double scales are dangerous. *The American Statistician*, 2(3):24, 1948.

[18] James W. Hardin and Joseph Hilbe. *Generalized Linear Models and Extensions*. Stata Press, 2001.

[19] James W. Hardin and Joseph Hilbe. *Generalized Estimating Equations*. Chapman & Hall/CRC, 2003.

[20] Markus Helbig and Simon Urbanek. *JGR—Java Gui for R*.
Available from http://www.rosuda.org/JGR, 2009. R package version 1.6-3.

[21] Joeseph Hilbe. *Logistic Regression Models*. Chapman & Hall/CRC Press, 2009.

[22] Joeseph Hilbe. *Creation of Synthetic Discrete Response Regression Models*. Stata Journal, 2010. forthcoming.

[23] Joseph Hilbe. *Negative Binomial Regression*. Cambridge University Press, 2007.

[24] Roger Koenker. *quantreg: Quantile Regression*.
Available from http://cran.r-project.org, 2009. R package version 4.38.

[25] Ulrich Kohler and Frauke Kreuter. *Data Analysis using Stata, 2nd ed.* Stata Press books. Stata Press, 2009.

[26] Max Kuhn and Steve Weaston. *odfWeave: Sweave processing of Open Document Format (ODF) files*.
Available from http://cran.r-project.org, 2009.

[27] Duncan Temple Lang, Debby Swayne, Hadley Wickham, and Michael Lawrence. *rggobi: Interface between R and GGobi*.
Available from http://www.ggobi.org/rggobi, 2008. R package version 2.1.10.

[28] Eric Lecoutre. The R2HTML package. *R News*, 3(3):33–36, 2003.
Available from http://cran.r-project.org.

[29] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, 2002.

[30] Jim Lemon and Philippe Grosjean. *prettyR: Pretty descriptive stats.* Available from http://cran.r-project.org, 2009.

[31] Thomas Lumley. *biglm: Bounded Memory Linear and Generalized Linear Models.*
Available from http://CRAN.R-project.org, 2009. R package version 0.7.

[32] David Meyer, Achim Zeileis, and Kurt Hornik. *vcd: Visualizing Categorical Data.*
Available from http://www.jstatsoft.org/v17/i03/, 2009. R package version 1.2-3.

[33] Michael N. Mitchell. *A Visual Guide to Stata Graphics, 2nd ed.* Stata Press, 2008.

[34] Robert A. Muenchen. *R for SAS and SPSS Users.* Springer, 2008.

[35] Paul Murrell. *R Graphics.* Chapman & Hall/CRC, 2005.

[36] Roger B. Newson. Rsource: Stata module to run r from inside stata using an r source file, 2008.
Available from http://ideas.repec.org/c/boc/bocode/s456847.html.

[37] Jose C. Pinheiro and Douglas M. Bates. *Mixed Effects Models in S and S-Plus.* Springer, 2002.

[38] R Development Core Team. *R: A Language and Environment for Statistical Computing.*
Available from http://www.R-project.org, 2008.

[39] R Development Core Team. *R Data Import/Export.*
Available from http://www.R-project.org, 2008.

[40] Deepayan Sarkar. *Lattice Multivariate Data Visualization with R.* Springer, 2007.

[41] Deepayan Sarkar. *lattice: Lattice Graphics.*
Available from http://CRAN.R-project.org, 2009. R package version 0.17-22.

[42] Gregory R. Warnes. Includes R source code and/or documentation contributed by Ben Bolker, Thomas Lumley, , and Randall C Johnson. *gmodels: Various R Programming Tools for Model Fitting.*
Available from http://CRAN.R-project.org, 2009. R package version 2.15.0.

[43] Gregory R. Warnes. Includes R source code and/or documentation contributed by (in alphabetical order) Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, Marc Schwartz, and Bill Venables. *gplots: Various R Programming Tools for Plotting Data.*
Available from http://CRAN.R-project.org, 2009. R package version 2.7.1.

[44] Phil Spector. *Data Manipulation with R (Use R)*. Springer, 2008.
[45] StataCorp. *Base Reference Manual*. Stata Press, 2009.
[46] StataCorp. *Data-Management Reference Manual*. Stata Press, 2009.
[47] Deborah F. Swayne, Duncan Temple Lang, Andreas Buja, and Dianne Cook. GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444, 2003.
[48] Duncan Temple Lang. The omegahat environment: New possibilities for statistical computing. *Journal of Computational and Graphical Statistics*, 9(3), 2000.
[49] S. Urbanek and M Theus. High interaction graphics for r, 2003.
Available from http://rosuda.org/iPlots/iplots.html.
[50] W. N. Venables. Exegeses on linear models, 1998.
Available from http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf.
[51] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S, 4th ed.* Springer, 2002.
[52] W. N. Venables, B. D. Ripley, and the R Core Development Team. *An Introduction to R*. Springer, 2007.
[53] Hadley Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12), 2007.
[54] Hadley Wickham. *ggplot2: An Implementation of the Grammar of Graphics*.
Available from http://cran.r-project.org, 2008. R package version 0.8.1.
[55] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.
[56] Leland Wilkinson, Graham Wills, Graham Wills, and Anonymous. *The Grammar of Graphics*. Springer, 2005.
[57] Graham Williams. *rattle: A Graphical User Interface for Data Mining in R Using GTK*. Available from http://rattle.togaware.com, 2008. R package version 2.3.128.

Index