

ADDISON
WESLEY
DATA &
ANALYTICS
SERIES

R

for Everyone

Advanced Analytics
and Graphics

JARED P. LANDER

FREE SAMPLE CHAPTER



SHARE WITH OTHERS

R for Everyone

The Addison-Wesley Data and Analytics Series



Visit informit.com/awdataseries for a complete list of available publications.

The Addison-Wesley Data and Analytics Series provides readers with practical knowledge for solving problems and answering questions with data. Titles in this series primarily focus on three areas:

1. **Infrastructure:** how to store, move, and manage data
2. **Algorithms:** how to mine intelligence or make predictions based on data
3. **Visualizations:** how to represent data and insights in a meaningful and compelling way

The series aims to tie all three of these areas together to help the reader build end-to-end systems for fighting spam; making recommendations; building personalization; detecting trends, patterns, or problems; and gaining insight from the data exhaust of systems and user interactions.



Make sure to connect with us!
informit.com/socialconnect

informit.com
the trusted technology learning source

◆ Addison-Wesley

Safari
Books Online

R for Everyone

Advanced Analytics and Graphics

Jared P. Lander

◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Lander, Jared P.

R for everyone / Jared P. Lander.

pages cm

Includes bibliographical references.

ISBN-13: 978-0-321-88803-7 (alk. paper)

ISBN-10: 0-321-88803-0 (alk. paper)

1. R (Computer program language) 2. Scripting languages (Computer science) 3. Statistics—Data processing. 4. Statistics—Graphic methods—Data processing. 5. Computer simulation. I. Title.

QA76.73.R3L36 2014

005.13—dc23

2013027407

Copyright © 2014 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-88803-7

ISBN-10: 0-321-88803-0

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.

First printing, December 2013



To my mother and father



This page intentionally left blank

Contents

Foreword xiii

Preface xv

Acknowledgments xix

About the Author xxiii

1 Getting R 1

- 1.1 Downloading R 1
- 1.2 R Version 2
- 1.3 32-bit versus 64-bit 2
- 1.4 Installing 2
- 1.5 Revolution R Community Edition 10
- 1.6 Conclusion 11

2 The R Environment 13

- 2.1 Command Line Interface 14
- 2.2 RStudio 15
- 2.3 Revolution Analytics RPE 26
- 2.4 Conclusion 27

3 R Packages 29

- 3.1 Installing Packages 29
- 3.2 Loading Packages 32
- 3.3 Building a Package 33
- 3.4 Conclusion 33

4 Basics of R 35

- 4.1 Basic Math 35
- 4.2 Variables 36
- 4.3 Data Types 38
- 4.4 Vectors 43
- 4.5 Calling Functions 49
- 4.6 Function Documentation 49
- 4.7 Missing Data 50
- 4.8 Conclusion 51

5 Advanced Data Structures 53

- 5.1 `data.frames` 53
- 5.2 Lists 61
- 5.3 Matrices 68
- 5.4 Arrays 71
- 5.5 Conclusion 72

6 Reading Data into R 73

- 6.1 Reading CSVs 73
- 6.2 Excel Data 74
- 6.3 Reading from Databases 75
- 6.4 Data from Other Statistical Tools 77
- 6.5 R Binary Files 77
- 6.6 Data Included with R 79
- 6.7 Extract Data from Web Sites 80
- 6.8 Conclusion 81

7 Statistical Graphics 83

- 7.1 Base Graphics 83
- 7.2 `ggplot2` 86
- 7.3 Conclusion 98

8 Writing R Functions 99

- 8.1 Hello, World! 99
- 8.2 Function Arguments 100
- 8.3 Return Values 103
- 8.4 `do.call` 104
- 8.5 Conclusion 104

9 Control Statements 105

- 9.1 `if` and `else` 105
- 9.2 `switch` 108
- 9.3 `ifelse` 109
- 9.4 Compound Tests 111
- 9.5 Conclusion 112

10 Loops, the Un-R Way to Iterate 113

- 10.1 `for` Loops 113
- 10.2 `while` Loops 115

10.3	Controlling Loops	115
10.4	Conclusion	116
11	Group Manipulation	117
11.1	Apply Family	117
11.2	aggregate	120
11.3	plyr	124
11.4	data.table	129
11.5	Conclusion	139
12	Data Reshaping	141
12.1	cbind and rbind	141
12.2	Joins	142
12.3	reshape2	149
12.4	Conclusion	153
13	Manipulating Strings	155
13.1	paste	155
13.2	sprintf	156
13.3	Extracting Text	157
13.4	Regular Expressions	161
13.5	Conclusion	169
14	Probability Distributions	171
14.1	Normal Distribution	171
14.2	Binomial Distribution	176
14.3	Poisson Distribution	182
14.4	Other Distributions	185
14.5	Conclusion	186
15	Basic Statistics	187
15.1	Summary Statistics	187
15.2	Correlation and Covariance	191
15.3	T-Tests	200
15.4	ANOVA	207
15.5	Conclusion	210

16	Linear Models	211
16.1	Simple Linear Regression	211
16.2	Multiple Regression	216
16.3	Conclusion	232
17	Generalized Linear Models	233
17.1	Logistic Regression	233
17.2	Poisson Regression	237
17.3	Other Generalized Linear Models	240
17.4	Survival Analysis	240
17.5	Conclusion	245
18	Model Diagnostics	247
18.1	Residuals	247
18.2	Comparing Models	253
18.3	Cross-Validation	257
18.4	Bootstrap	262
18.5	Stepwise Variable Selection	265
18.6	Conclusion	269
19	Regularization and Shrinkage	271
19.1	Elastic Net	271
19.2	Bayesian Shrinkage	290
19.3	Conclusion	295
20	Nonlinear Models	297
20.1	Nonlinear Least Squares	297
20.2	Splines	300
20.3	Generalized Additive Models	304
20.4	Decision Trees	310
20.5	Random Forests	312
20.6	Conclusion	313
21	Time Series and Autocorrelation	315
21.1	Autoregressive Moving Average	315
21.2	VAR	322

21.3	GARCH	327
21.4	Conclusion	336
22	Clustering	337
22.1	K-means	337
22.2	PAM	345
22.3	Hierarchical Clustering	352
22.4	Conclusion	357
23	Reproducibility, Reports and Slide Shows with knitr	359
23.1	Installing a \LaTeX Program	359
23.2	\LaTeX Primer	360
23.3	Using knitr with \LaTeX	362
23.4	Markdown Tips	367
23.5	Using knitr and Markdown	368
23.6	pandoc	369
23.7	Conclusion	371
24	Building R Packages	373
24.1	Folder Structure	373
24.2	Package Files	373
24.3	Package Documentation	380
24.4	Checking, Building and Installing	383
24.5	Submitting to CRAN	384
24.6	C++ Code	384
24.7	Conclusion	390
A	Real-Life Resources	391
A.1	Meetups	391
A.2	Stackoverflow	392
A.3	Twitter	393
A.4	Conferences	393
A.5	Web Sites	393
A.6	Documents	394
A.7	Books	394
A.8	Conclusion	394

B	Glossary	395
	List of Figures	409
	List of Tables	417
	General Index	419
	Index of Functions	427
	Index of Packages	431
	Index of People	433
	Data Index	435

Foreword

R has had tremendous growth in popularity over the last three years. Based on that, you'd think that it was a new, up-and-coming language. But surprisingly, R has been around since 1993. Why the sudden uptick in popularity? The somewhat obvious answer seems to be the emergence of data science as a career and a field of study. But the underpinnings of data science have been around for many decades. Statistics, linear algebra, operations research, artificial intelligence, and machine learning all contribute parts to the tools that a modern data scientist uses. R, more than most languages, has been built to make most of these tools only a single function call away.

That's why I'm very excited to have this book as one of the first in the Addison-Wesley Data and Analytics Series. R is indispensable for many data science tasks. Many algorithms useful for prediction and analysis can be accessed through only a few lines of code, which makes it a great fit for solving modern data challenges. Data science as a field isn't just about math and statistics, and it isn't just about programming and infrastructure. This book provides a well-balanced introduction to the power and expressiveness of R and is aimed at a general audience.

I can't think of a better author to provide an introduction to R than Jared Lander. Jared and I first met through the New York City machine learning community in late 2009. Back then, the New York City data community was small enough to fit in a single conference room, and many of the other data meetups had yet to be formed. Over the last four years, Jared has been at the forefront of the emerging data science profession.

Through running the Open Statistical Programming Meetup, speaking at events, and teaching a course at Columbia on R, Jared has helped grow the community by educating programmers, data scientists, journalists, and statisticians alike. But Jared's expertise isn't limited to teaching. As an everyday practitioner, he puts these tools to use while consulting for clients big and small.

This book provides an introduction both to programming in R and to the various statistical methods and tools an everyday R programmer uses. Examples use publicly available datasets that Jared has helpfully cleaned and made accessible through his Web site. By using real data and setting up interesting problems, this book stays engaging to the end.

—Paul Dix, *Series Editor*

This page intentionally left blank

Preface

With the increasing prevalence of data in our daily lives, new and better tools are needed to analyze the deluge. Traditionally there have been two ends of the spectrum: lightweight, individual analysis using tools like Excel or SPSS and heavy duty, high-performance analysis built with C++ and the like. With the increasing strength of personal computers grew a middle ground that was both interactive and robust. Analysis done by an individual on his or her own computer in an exploratory fashion could quickly be transformed into something destined for a server, underpinning advanced business processes. This area is the domain of R, Python, and other scripted languages.

R, invented by Robert Gentleman and Ross Ihaka of the University of Auckland in 1993, grew out of S, which was invented by John Chambers at Bell Labs. It is a high-level language that was originally intended to be run interactively where the user runs a command, gets a result, and then runs another command. It has since evolved into a language that can also be embedded in systems and tackle complex problems.

In addition to transforming and analyzing data, R can produce amazing graphics and reports with ease. It is now being used as a full stack for data analysis, extracting and transforming data, fitting models, drawing inferences and making predictions, plotting and reporting results.

R's popularity has skyrocketed since the late 2000s, as it has stepped out of academia and into banking, marketing, pharmaceuticals, politics, genomics and many other fields. Its new users are often shifting from low-level, compiled languages like C++, other statistical packages such as SAS or SPSS, and from the 800-pound gorilla, Excel. This time period also saw a rapid surge in the number of add-on packages—libraries of prewritten code that extend R's functionality.

While R can sometimes be intimidating to beginners, especially for those without programming experience, I find that programming analysis, instead of pointing and clicking, soon becomes much easier, more convenient and more reliable. It is my goal to make that learning process easier and quicker.

This book lays out information in a way I wish I were taught when learning R in graduate school. Coming full circle, the content of this book was developed in conjunction with the data science course I teach at Columbia University. It is not meant to cover every minute detail of R, but rather the 20% of functionality needed to accomplish 80% of the work. The content is organized into self-contained chapters as follows.

Chapter 1, Getting R: Where to download R and how to install it. This deals with the varying operating systems and 32-bit versus 64-bit versions. It also gives advice on where to install R.

Chapter 2, The R Environment: An overview of using R, particularly from within RStudio. RStudio projects and Git integration are covered as is customizing and navigating RStudio.

Chapter 3, Packages: How to locate, install and load R packages.

Chapter 4, Basics of R: Using R for math. Variable types such as `numeric`, `character` and `Date` are detailed as are `vectors`. There is a brief introduction to calling functions and finding documentation on functions.

Chapter 5, Advanced Data Structures: The most powerful and commonly used data structure, `data.frames`, along with `matrices` and `lists`, are introduced.

Chapter 6, Reading Data into R: Before data can be analyzed it must be read into R. There are numerous ways to ingest data, including reading from CSVs and databases.

Chapter 7, Statistical Graphics: Graphics are a crucial part of preliminary data analysis and communicating results. R can make beautiful plots using its powerful plotting utilities. Base graphics and `ggplot2` are introduced and detailed here.

Chapter 8, Writing R Functions: Repeatable analysis is often made easier with user-defined functions. The structure, arguments and return rules are discussed.

Chapter 9, Control Statements: Controlling the flow of programs using `if`, `ifelse` and complex checks.

Chapter 10, Loops, the Un-R Way to Iterate: Iterating using `for` and `while` loops. While these are generally discouraged they are important to know.

Chapter 11, Group Manipulation: A better alternative to loops, vectorization does not quite iterate through data so much as operate on all elements at once. This is more efficient and is primarily performed with the `apply` functions and `plyr` package.

Chapter 12, Data Reshaping: Combining multiple datasets, whether by stacking or joining, is commonly necessary as is changing the shape of data. The `plyr` and `reshape2` packages offer good functions for accomplishing this in addition to base tools such as `rbind`, `cbind` and `merge`.

Chapter 13, Manipulating Strings: Most people do not associate character data with statistics but it is an important form of data. R provides numerous facilities for working with strings, including combining them and extracting information from within. Regular expressions are also detailed.

Chapter 14, Probability Distributions: A thorough look at the normal, binomial and Poisson distributions. The formulas and functions for many distributions are noted.

Chapter 15, Basic Statistics: These are the first statistics most people are taught, such as mean, standard deviation and t-tests.

Chapter 16, Linear Models: The most powerful and common tool in statistics, linear models are extensively detailed.

Chapter 17, Generalized Linear Models: Linear models are extended to include logistic and Poisson regression. Survival analysis is also covered.

Chapter 18, Model Diagnostics: Determining the quality of models and variable selection using residuals, AIC, cross-validation, the bootstrap and stepwise variable selection.

Chapter 19, Regularization and Shrinkage: Preventing overfitting using the Elastic Net and Bayesian methods.

Chapter 20, Nonlinear Models: When linear models are inappropriate, nonlinear models are a good solution. Nonlinear least squares, splines, generalized additive models, decision trees and random forests are discussed.

Chapter 21, Time Series and Autocorrelation: Methods for the analysis of univariate and multivariate time series data.

Chapter 22, Clustering: Clustering, the grouping of data, is accomplished by various methods such as K-means and hierarchical clustering.

Chapter 23, Reproducibility, Reports and Slide Shows with `knitr`: Generating reports, slide shows and Web pages from within R is made easy with `knitr`, \LaTeX and Markdown.

Chapter 24, Building R Packages: R packages are great for portable, reusable code. Building these packages has been made incredibly easy with the advent of `devtools` and `Rcpp`.

Appendix A, Real-Life Resources: A listing of our favorite resources for learning more about R and interacting with the community.

Appendix B, Glossary: A glossary of terms used throughout this book.

A good deal of the text in this book is either R code or the results of running code. Code and results are most often in a separate block of text and set in a distinctive font, as shown in the following example. The different parts of code also have different colors. Lines of code start with `>`, and if code is continued from one line to another the continued line begins with `+`.

```
> # this is a comment
>
> # now basic math
> 10 * 10

[1] 100

>
> # calling a function
> sqrt(4)

[1] 2
```

Certain Kindle devices do not display color so the digital edition of this book will be viewed in grayscale on those devices.

There are occasions where code is shown inline and looks like `sqrt(4)`.

In the few places where math is necessary, the equations are indented from the margin and are numbered.

$$e^{i\pi} + 1 = 0 \tag{1}$$

Within equations, normal variables appear as italic text (x), vectors are bold lowercase letters (\mathbf{x}) and matrices are bold uppercase letters (\mathbf{X}). Greek letters, such as α and β , follow the same convention.

Function names will be written as `join` and package names as `plyr`. Objects generated in code that are referenced in text are written as `object1`.

Learning R is a gratifying experience that makes life so much easier for so many tasks. I hope you enjoy learning with me.

Acknowledgments

To start, I must thank my mother, Gail Lander, for encouraging me to become a math major. Without that I would never have followed the path that led me to statistics and data science. In a similar vein, I have to thank my father, Howard Lander, for paying all those tuition bills. He has been a valuable source of advice and guidance throughout my life and someone I have aspired to emulate in many ways. While they both insist they do not understand what I do, they love that I do it and have helped me all along the way. Staying with family, I should thank my sister and brother-in-law, Aimee and Eric Schechterman, for letting me teach math to Noah, their five-year-old son.

There are many teachers who have helped shape me over the years. The first is Rochelle Lecke, who tutored me in middle school math even when my teacher told me I did not have worthwhile math skills.

Then there is Beth Edmondson, my precalc teacher at Princeton Day School. After I wasted the first half of high school as a mediocre student, she told me I had “some nerve signing up for next year’s AP Calc” given my grades. She agreed to let me take AP Calc if I went from a C to an A+ in her class, never thinking I stood a chance. Three months later, she was in shock as I not only earned the A+, but turned around my entire academic career. She changed my life and without her, I do not know where I would be today. I am forever grateful that she was my teacher.

For the first two years at Muhlenberg College, I was determined to be a business and communications major, but took math classes because they came naturally to me. My professors, Dr. Penny Dunham, Dr. Bill Dunham, and Dr. Linda McGuire, all convinced me to become a math major, a decision that has greatly shaped my life. Dr. Greg Cicconetti gave me my first glimpse of rigorous statistics, my first research opportunity and planted the idea in my head that I should go to grad school for statistics.

While earning my M.A. at Columbia University, I was surrounded by brilliant minds in statistics and programming. Dr. David Madigan opened my eyes to modern machine learning, and Dr. Bodhi Sen got me thinking about statistical programming. I had the privilege to do research with Dr. Andrew Gelman, whose insights have been immeasurably important to me. Dr. Richard Garfield showed me how to use statistics to help people in disaster and war zones when he sent me on my first assignment to Myanmar. His advice and friendship over the years have been dear to me. Dr. Jingchen Liu

allowed and encouraged me to write my thesis on New York City pizza, which has brought me an inordinate amount of attention.¹

While at Columbia, I also met my good friend—and one time TA— Dr. Ivor Cribben who filled in so many gaps in my knowledge. Through him, I met Dr. Rachel Schutt, a source of great advice, and who I am now honored to teach alongside at Columbia.

Grad school might never have happened without the encouragement and support of Shanna Lee. She helped maintain my sanity while I was incredibly overcommitted to two jobs, classes and Columbia's hockey team. I am not sure I would have made it through without her.

Steve Czetty gave me my first job in analytics at Sky IT Group and taught me about databases, while letting me experiment with off-the-wall programming. This sparked my interest in statistics and data. Joe DeSiena, Philip du Plessis, and Ed Bobrin at the Bardess Group are some of the finest people I have ever had the pleasure to work with, and I am proud to be working with them to this day. Mike Minelli, Rich Kittler, Mark Barry, David Smith, Joseph Rickert, Dr. Norman Nie, James Peruvankal, Neera Talbert and Dave Rich at Revolution Analytics let me do one of the best jobs I could possibly imagine: explaining to people in business why they should be using R. Kirk Mettler, Richard Schultz, Dr. Bryan Lewis and Jim Winfield at Big Computing encouraged me to have fun, tackling interesting problems in R. Vincent Saulys, John Weir, and Dr. Saar Golde at Goldman Sachs made my time there both enjoyable and educational.

Throughout the course of writing this book, many people helped me with the process. First and foremost is Yin Cheung, who saw all the stress I constantly felt and supported me through many ruined nights and days.

My editor, Debra Williams, knew just how to encourage me and her guiding hand has been invaluable. Paul Dix, the series editor and a good friend, was the person who suggested I write this book, so none of this would have happened without him. Thanks to Caroline Senay and Andrea Fox for being great copy editors. Without them, this book would not be nearly as well put together. Robert Mauriello's technical review was incredibly useful in honing the book's presentation.

The folks at RStudio, particularly JJ Allaire and Josh Paulson, make an amazing product, which made the writing process far easier than it would have been otherwise. Yihui Xie, the author of the `knitr` package, provided numerous feature changes that I needed to write this book. His software, and his speed at implementing my requests, is greatly appreciated.

Numerous people have provided valuable feedback as I produced this book, including Chris Bethel, Dr. Dirk Eddebuettel, Dr. Ramnath Vaidyanathan, Dr. Eran Bellin,

1. <http://slice.seriousseats.com/archives/2010/03/the-moneyball-of-pizza-statistician-uses-statistics-to-find-nyc-best-pizza.html>

Avi Fisher, Brian Ezra, Paul Puglia, Nicholas Galasinao, Aaron Schumaker, Adam Hogan, Jeffrey Arnold, and John Houston.

Last fall was my first time teaching, and I am thankful to the students from the Fall 2012 Introduction to Data Science class at Columbia University for being the guinea pigs for the material that ultimately ended up in this book.

Thank you to everyone who helped along the way.

This page intentionally left blank

About the Author

Jared P. Lander is the founder and CEO of Lander Analytics, a statistical consulting firm based in New York City, the organizer of the New York Open Statistical Programming Meetup, and an adjunct professor of statistics at Columbia University. He is also a tour guide for Scott's Pizza Tours and an advisor to Brewla Bars, a gourmet ice pop start-up. With an M.A. from Columbia University in statistics and a B.A. from Muhlenberg College in mathematics, he has experience in both academic research and industry. His work for both large and small organizations spans politics, tech start-ups, fund-raising, music, finance, healthcare and humanitarian relief efforts.

He specializes in data management, multilevel models, machine learning, generalized linear models, visualization, data management and statistical computing.

This page intentionally left blank

This page intentionally left blank

Chapter 12

Data Reshaping

As noted in Chapter 11, manipulating the data takes a great deal of effort before serious analysis can begin. In this chapter we will consider when the data needs to be rearranged from column oriented to row oriented (or the opposite) and when the data are in multiple, separate sets and need to be combined into one.

There are base functions to accomplish these tasks but we will focus on those in `plyr`, `reshape2` and `data.table`.

12.1 `cbind` and `rbind`

The simplest case is when we have two datasets with either identical columns (both the number of and names) or the same number of rows. In this case, either `rbind` or `cbind` work great.

As a first trivial example, we create two simple `data.frames` by combining a few vectors with `cbind`, and then stack them using `rbind`.

```
> # make two vectors and combine them as columns in a data.frame
> sport <- c("Hockey", "Baseball", "Football")
> league <- c("NHL", "MLB", "NFL")
> trophy <- c("Stanley Cup", "Commissioner's Trophy",
+           "Vince Lombardi Trophy")
> trophies1 <- cbind(sport, league, trophy)
> # make another data.frame using data.frame()
> trophies2 <- data.frame(sport=c("Basketball", "Golf"),
+                         league=c("NBA", "PGA"),
+                         trophy=c("Larry O'Brien Championship Trophy",
+                                 "Wanamaker Trophy"),
+                         stringsAsFactors=FALSE)
> # combine them into one data.frame with rbind
> trophies <- rbind(trophies1, trophies2)
```

Both `cbind` and `rbind` can take multiple arguments to combine an arbitrary number of objects. Note that it is possible to assign new column names to vectors in `cbind`.

```
> cbind(Sport = sport, Association = league, Prize = trophy)

      Sport      Association Prize
[1,] "Hockey"    "NHL"      "Stanley Cup"
[2,] "Baseball"  "MLB"      "Commissioner's Trophy"
[3,] "Football"  "NFL"      "Vince Lombardi Trophy"
```

12.2 Joins

Data do not always come so nicely aligned for combining using `cbind`, so they need to be joined together using a common key. This concept should be familiar to SQL users. Joins in R are not as flexible as SQL joins, but are still an essential operation in the data analysis process.

The three most commonly used functions for joins are `merge` in base R, `join` in `plyr` and the merging functionality in `data.table`. Each has pros and cons with some pros outweighing their respective cons.

To illustrate these functions I have prepared data originally made available as part of the USAID Open Government initiative.¹ The data have been chopped into eight separate files so that they can be joined together. They are all available in a zip file at http://jaredlander.com/data/US_Foreign_Aid.zip. These should be downloaded and unzipped to a folder on our computer. This can be done a number of ways (including using a mouse!) but we show how to download and unzip using R.

```
> download.file(url="http://jaredlander.com/data/US_Foreign_Aid.zip",
+              destfile="data/ForeignAid.zip")
> unzip("data/ForeignAid.zip", exdir="data")
```

To load all of these files programmatically, we use a `for` loop as seen in Section 10.1. We get a list of the files using `dir`, and then loop through that list assigning each dataset to a name specified using `assign`.

```
> require(stringr)
> # first get a list of the files
> theFiles <- dir("data/", pattern="\\.csv")
> ## loop through those files
> for(a in theFiles)
+ {
+   # build a good name to assign to the data
+   nameToUse <- str_sub(string=a, start=12, end=18)
```

1. More information about the data is available at <http://gbk.eads.usaidallnet.gov/>.

```
+ # read in the csv using read.table
+ # file.path is a convenient way to specify a folder and file name
+ temp <- read.table(file=file.path("data", a),
+                   header=TRUE, sep=",", stringsAsFactors=FALSE)
+ # assign them into the workspace
+ assign(x=nameToUse, value=temp)
+ }
```

12.2.1 merge

R comes with a built-in function, called `merge`, to merge two `data.frames`.

```
> Aid90s00s <- merge(x=Aid_90s, y=Aid_00s,
+                   by.x=c("Country.Name", "Program.Name"),
+                   by.y=c("Country.Name", "Program.Name"))
> head(Aid90s00s)
```

	Country.Name			Program.Name					
1	Afghanistan			Child Survival and Health					
2	Afghanistan			Department of Defense Security Assistance					
3	Afghanistan			Development Assistance					
4	Afghanistan			Economic Support Fund/Security Support Assistance					
5	Afghanistan			Food For Education					
6	Afghanistan			Global Health and Child Survival					
	FY1990	FY1991	FY1992	FY1993	FY1994	FY1995	FY1996	FY1997	FY1998
1	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	NA	NA	NA	14178135	2769948	NA	NA	NA	NA
5	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA	NA	NA	NA
	FY1999	FY2000	FY2001	FY2002	FY2003	FY2004		FY2005	
1	NA	NA	NA	2586555	56501189	40215304		39817970	
2	NA	NA	NA	2964313	NA	45635526		151334908	
3	NA	NA	4110478	8762080	54538965	180539337		193598227	
4	NA	NA	61144	31827014	341306822	1025522037		1157530168	
5	NA	NA	NA	NA	3957312	2610006		3254408	
6	NA	NA	NA	NA	NA	NA		NA	
	FY2006		FY2007	FY2008	FY2009				
1	40856382		72527069	28397435	NA				
2	230501318		214505892	495539084	552524990				
3	212648440		173134034	150529862	3675202				
4	1357750249		1266653993	1400237791	1418688520				
5	386891		NA	NA	NA				
6	NA		NA	63064912	1764252				

The `by.x` specifies the key column(s) in the left `data.frame` and `by.y` does the same for the right `data.frame`. The ability to specify different column names for each `data.frame` is the most useful feature of `merge`. The biggest drawback, however, is that `merge` can be much slower than the alternatives.

12.2.2 `plyr` join

Returning to Hadley Wickham's `plyr` package, we see it includes a `join` function, which works similarly to `merge` but is much faster. The biggest drawback, though, is that the key column(s) in each table must have the same name. We use the same data used previously to illustrate.

```
> require(plyr)
> Aid90s00sJoin <- join(x = Aid_90s, y = Aid_00s, by = c("Country.Name",
+           "Program.Name"))
> head(Aid90s00sJoin)
```

	Country.Name			Program.Name						
1	Afghanistan			Child Survival and Health						
2	Afghanistan			Department of Defense Security Assistance						
3	Afghanistan			Development Assistance						
4	Afghanistan			Economic Support Fund/Security Support Assistance						
5	Afghanistan			Food For Education						
6	Afghanistan			Global Health and Child Survival						
	FY1990	FY1991	FY1992	FY1993	FY1994	FY1995	FY1996	FY1997	FY1998	
1	NA	NA	NA	NA	NA	NA	NA	NA	NA	
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	
3	NA	NA	NA	NA	NA	NA	NA	NA	NA	
4	NA	NA	NA	14178135	2769948	NA	NA	NA	NA	
5	NA	NA	NA	NA	NA	NA	NA	NA	NA	
6	NA	NA	NA	NA	NA	NA	NA	NA	NA	
	FY1999	FY2000	FY2001	FY2002	FY2003	FY2004	FY2005			
1	NA	NA	NA	2586555	56501189	40215304	39817970			
2	NA	NA	NA	2964313	NA	45635526	151334908			
3	NA	NA	4110478	8762080	54538965	180539337	193598227			
4	NA	NA	61144	31827014	341306822	1025522037	1157530168			
5	NA	NA	NA	NA	3957312	2610006	3254408			
6	NA	NA	NA	NA	NA	NA	NA			
	FY2006	FY2007	FY2008	FY2009						
1	40856382	72527069	28397435	NA						
2	230501318	214505892	495539084	552524990						
3	212648440	173134034	150529862	3675202						
4	1357750249	1266653993	1400237791	1418688520						
5	386891	NA	NA	NA						
6	NA	NA	63064912	1764252						

`join` has an argument for specifying a left, right, inner or full (outer) join.

We have eight `data.frames` containing foreign assistance data that we would like to combine into one `data.frame` without hand coding each join. The best way to do this is to put all the `data.frames` into a `list`, and then successively join them together using `Reduce`.

```
> # first figure out the names of the data.frames
> frameNames <- str_sub(string = theFiles, start = 12, end = 18)
> # build an empty list
> frameList <- vector("list", length(frameNames))
> names(frameList) <- frameNames
> # add each data.frame into the list
> for (a in frameNames)
+ {
+   frameList[[a]] <- eval(parse(text = a))
+ }
```

A lot happened in that section of code, so let's go over it carefully. First we reconstructed the names of the `data.frames` using `str_sub` from Hadley Wickham's `stringr` package, which is shown in more detail in Chapter 13. Then we built an empty `list` with as many elements as there are `data.frames`, in this case eight, using `vector` and assigning its mode to "list." We then set appropriate names to the `list`.

Now that the `list` is built and named, we loop through it, assigning to each element the appropriate `data.frame`. The problem is that we have the names of the `data.frames` as characters but the `<-` operator requires a variable, not a character. So we `parse` and `evaluate` the character, which realizes the actual variable. Inspecting, we see that the `list` does indeed contain the appropriate `data.frames`.

```
> head(frameList[[1]])
```

	Country.Name		Program.Name				
1	Afghanistan		Child Survival and Health				
2	Afghanistan		Department of Defense Security Assistance				
3	Afghanistan		Development Assistance				
4	Afghanistan		Economic Support Fund/Security Support Assistance				
5	Afghanistan		Food For Education				
6	Afghanistan		Global Health and Child Survival				
	FY2000	FY2001	FY2002	FY2003	FY2004	FY2005	FY2006
1	NA	NA	2586555	56501189	40215304	39817970	40856382
2	NA	NA	2964313	NA	45635526	45635526	230501318
3	NA	4110478	8762080	54538965	180539337	193598227	212648440
4	NA	61144	31827014	341306822	1025522037	1157530168	1357750249
5	NA	NA	NA	3957312	2610006	3254408	386891
6	NA	NA	NA	NA	NA	NA	NA

```

      FY2007      FY2008      FY2009
1  72527069  28397435      NA
2  214505892 495539084  552524990
3  173134034 150529862   3675202
4 1266653993 1400237791 1418688520
5      NA      NA      NA
6      NA  63064912  1764252

```

```
> head(frameList[["Aid_00s"]])
```

```

Country.Name      Program.Name
1 Afghanistan      Child Survival and Health
2 Afghanistan      Department of Defense Security Assistance
3 Afghanistan      Development Assistance
4 Afghanistan Economic Support Fund/Security Support Assistance
5 Afghanistan      Food For Education
6 Afghanistan      Global Health and Child Survival
FY2000  FY2001  FY2002  FY2003  FY2004  FY2005  FY2006
1      NA      NA  2586555  56501189  40215304  39817970  40856382
2      NA      NA  2964313      NA  45635526  151334908  230501318
3      NA 4110478  8762080  54538965  180539337  193598227  212648440
4      NA  61144 31827014 341306822 1025522037 1157530168 1357750249
5      NA      NA      NA  3957312  2610006  3254408  386891
6      NA      NA      NA      NA      NA      NA      NA
      FY2007      FY2008      FY2009
1  72527069  28397435      NA
2  214505892 495539084  552524990
3  173134034 150529862   3675202
4 1266653993 1400237791 1418688520
5      NA      NA      NA
6      NA  63064912  1764252

```

```
> head(frameList[[5]])
```

```

Country.Name      Program.Name
1 Afghanistan      Child Survival and Health
2 Afghanistan      Department of Defense Security Assistance
3 Afghanistan      Development Assistance
4 Afghanistan Economic Support Fund/Security Support Assistance
5 Afghanistan      Food For Education
6 Afghanistan      Global Health and Child Survival
FY1960  FY1961  FY1962  FY1963  FY1964  FY1965  FY1966  FY1967  FY1968
1      NA      NA      NA      NA      NA      NA      NA      NA      NA
2      NA      NA      NA      NA      NA      NA      NA      NA      NA
3      NA      NA      NA      NA      NA      NA      NA      NA      NA

```



```

4    NA    NA 181177853    NA    NA    NA    NA    NA    NA
5    NA    NA    NA    NA    NA    NA    NA    NA    NA
6    NA    NA    NA    NA    NA    NA    NA    NA    NA

```

```
FY1969
```

```

1    NA
2    NA
3    NA
4    NA
5    NA
6    NA

```

```
> head(frameList[["Aid_60s"]])
```

```

Country.Name          Program.Name
1 Afghanistan          Child Survival and Health
2 Afghanistan      Department of Defense Security Assistance
3 Afghanistan          Development Assistance
4 Afghanistan Economic Support Fund/Security Support Assistance
5 Afghanistan          Food For Education
6 Afghanistan          Global Health and Child Survival

```

```

FY1960 FY1961    FY1962 FY1963 FY1964 FY1965 FY1966 FY1967 FY1968
1    NA    NA    NA    NA    NA    NA    NA    NA    NA
2    NA    NA    NA    NA    NA    NA    NA    NA    NA
3    NA    NA    NA    NA    NA    NA    NA    NA    NA
4    NA    NA 181177853    NA    NA    NA    NA    NA    NA
5    NA    NA    NA    NA    NA    NA    NA    NA    NA
6    NA    NA    NA    NA    NA    NA    NA    NA    NA

```

```
FY1969
```

```

1    NA
2    NA
3    NA
4    NA
5    NA
6    NA

```

Having all the data.frames in a list allows us to iterate through the list, joining all the elements together (or applying any function to the elements iteratively). Rather than using a loop, we use the Reduce function to speed up the operation.

```

> allAid <- Reduce(function(...)
+ {
+   join(..., by = c("Country.Name", "Program.Name"))
+ }, frameList)
> dim(allAid)

```

```
[1] 2453 67
```

```

> require(reshape2)
> corner(allAid, c = 15)

Country.Name          Program.Name
1 Afghanistan        Child Survival and Health
2 Afghanistan        Department of Defense Security Assistance
3 Afghanistan        Development Assistance
4 Afghanistan Economic Support Fund/Security Support Assistance
5 Afghanistan        Food For Education
  FY2000  FY2001  FY2002  FY2003  FY2004  FY2005  FY2006
1      NA      NA  2586555  56501189  40215304  39817970  40856382
2      NA      NA  2964313      NA  45635526  151334908  230501318
3      NA  4110478  8762080  54538965  180539337  193598227  212648440
4      NA   61144  31827014  341306822  1025522037  1157530168  1357750249
5      NA      NA      NA   3957312   2610006   3254408   386891
  FY2007  FY2008  FY2009  FY2010  FY1946  FY1947
1  72527069  28397435      NA      NA      NA      NA
2  214505892  495539084  552524990  316514796      NA      NA
3  173134034  150529862   3675202      NA      NA      NA
4  1266653993  1400237791  1418688520  2797488331      NA      NA
5           NA           NA           NA           NA      NA      NA

> bottomleft(allAid, c = 15)

Country.Name          Program.Name  FY2000  FY2001  FY2002
2449 Zimbabwe Other State Assistance  1341952  322842      NA
2450 Zimbabwe Other USAID Assistance  3033599  8464897  6624408
2451 Zimbabwe Peace Corps           2140530  1150732  407834
2452 Zimbabwe Title I                NA      NA      NA
2453 Zimbabwe Title II               NA      NA  31019776
  FY2003  FY2004  FY2005  FY2006  FY2007  FY2008  FY2009
2449      NA  318655  44553  883546  1164632  2455592  2193057
2450 11580999 12805688 10091759 4567577 10627613 11466426 41940500
2451      NA      NA      NA      NA      NA      NA      NA
2452      NA      NA      NA      NA      NA      NA      NA
2453      NA      NA      NA  277468 100053600 180000717 174572685
  FY2010  FY1946  FY1947
2449 1605765      NA      NA
2450 30011970      NA      NA
2451      NA      NA      NA
2452      NA      NA      NA
2453 79545100      NA      NA

```

Reduce can be a difficult function to grasp, so we illustrate it with a simple example. Let's say we have a vector of the first ten integers, 1:10, and want to sum them (forget for a moment that `sum(1:10)` will work perfectly). We can call `Reduce(sum, 1:10)`,

which will first add 1 and 2. It will then add 3 to that result, then 4 to that result, and so on, resulting in 55.

Likewise, we passed a `list` to a function that joins its inputs, which in this case was simply `...`, meaning that anything could be passed. Using `...` is an advanced trick of R programming that can be difficult to get right. `Reduce` passed the first two `data.frames` in the `list`, which were then joined. That result was then joined to the next `data.frame` and so on until they were all joined together.

12.2.3 data.table merge

Like many other operations in `data.table`, joining data requires a different syntax, and possibly a different way of thinking. To start, we convert two of our foreign aid datasets' `data.frames` into `data.tables`.

```
> require(data.table)
> dt90 <- data.table(Aid_90s, key = c("Country.Name", "Program.Name"))
> dt00 <- data.table(Aid_00s, key = c("Country.Name", "Program.Name"))
```

Then, doing the join is a simple operation. Note that the join requires specifying the keys for the `data.tables`, which we did during their creation.

```
> dt0090 <- dt90[dt00]
```

In this case `dt90` is the left side, `dt00` is the right side and a left join was performed.

12.3 reshape2

The next most common munging need is either melting data (going from column orientation to row orientation) or casting data (going from row orientation to column orientation). As with most other procedures in R, there are multiple functions available to accomplish these tasks but we will focus on Hadley Wickham's `reshape2` package. (We talk about Wickham a lot because his products have become so fundamental to the R developer's toolbox.)

12.3.1 melt

Looking at the `Aid_00s` `data.frame`, we see that each year is stored in its own column. That is, the dollar amount for a given country and program is found in a different column for each year. This is called a cross table, which, while nice for human consumption, is not ideal for graphing with `ggplot2` or for some analysis algorithms.

```
> head(Aid_00s)
```

	Country.Name	Program.Name
1	Afghanistan	Child Survival and Health
2	Afghanistan	Department of Defense Security Assistance

```

3 Afghanistan Development Assistance
4 Afghanistan Economic Support Fund/Security Support Assistance
5 Afghanistan Food For Education
6 Afghanistan Global Health and Child Survival
  FY2000 FY2001 FY2002 FY2003 FY2004 FY2005 FY2006
1 NA NA 2586555 56501189 40215304 39817970 40856382
2 NA NA 2964313 NA 45635526 151334908 230501318
3 NA 4110478 8762080 54538965 180539337 193598227 212648440
4 NA 61144 31827014 341306822 1025522037 1157530168 1357750249
5 NA NA NA 3957312 2610006 3254408 386891
6 NA NA NA NA NA NA NA
  FY2007 FY2008 FY2009
1 72527069 28397435 NA
2 214505892 495539084 552524990
3 173134034 150529862 3675202
4 1266653993 1400237791 1418688520
5 NA NA NA
6 NA 63064912 1764252

```

We want it set up so that each row represents a single country-program-year entry with the dollar amount stored in one column. To achieve this we melt the data using `melt` from `reshape2`.

```

> require(reshape2)
> melt00 <- melt(Aid_00s, id.vars=c("Country.Name", "Program.Name"),
+               variable.name="Year", value.name="Dollars")
> tail(melt00, 10)

```

```

Country.Name
24521 Zimbabwe
24522 Zimbabwe
24523 Zimbabwe
24524 Zimbabwe
24525 Zimbabwe
24526 Zimbabwe
24527 Zimbabwe
24528 Zimbabwe
24529 Zimbabwe
24530 Zimbabwe

Program.Name Year
24521 Migration and Refugee Assistance FY2009
24522 Narcotics Control FY2009

```

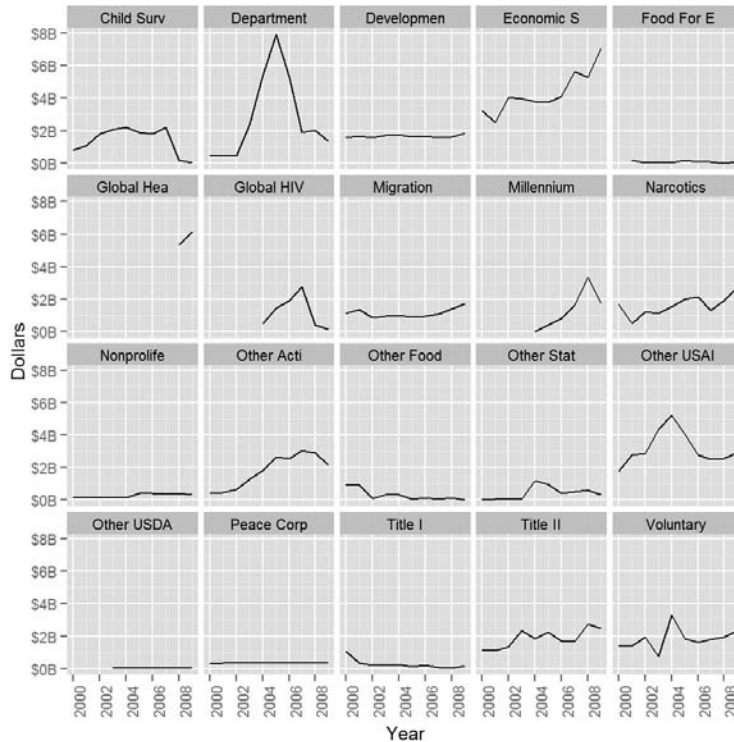



Figure 12.1 Plot of foreign assistance by year for each of the programs.

12.3.2 dcast

Now that we have the foreign aid data melted, we cast it back into the wide format for illustration purposes. The function for this is `dcast`, and it has trickier arguments than `melt`. The first is the data to be used, in our case `melt00`. The second argument is a formula where the left side specifies the columns that should remain columns and the right side specifies the columns that should become row names. The third argument is the column (as a character) that holds the values to be populated into the new columns representing the unique values of the right side of the formula argument.

```
> cast00 <- dcast(melt00, Country.Name + Program.Name ~ Year,
+               value.var = "Dollars")
> head(cast00)
```

Country.Name	Program.Name	2000
1 Afghanistan	Child Survival and Health	NA
2 Afghanistan	Department of Defense Security Assistance	NA
3 Afghanistan	Development Assistance	NA

```

4 Afghanistan Economic Support Fund/Security Support Assistance NA
5 Afghanistan Food For Education NA
6 Afghanistan Global Health and Child Survival NA
  2001      2002      2003      2004      2005      2006
1    NA 2586555 56501189 40215304 39817970 40856382
2    NA 2964313          NA 45635526 151334908 230501318
3 4110478 8762080 54538965 180539337 193598227 212648440
4   61144 31827014 341306822 1025522037 1157530168 1357750249
5    NA      NA 3957312 2610006 3254408 386891
6    NA      NA      NA      NA      NA      NA
  2007      2008      2009
1 72527069 28397435      NA
2 214505892 495539084 552524990
3 173134034 150529862 3675202
4 1266653993 1400237791 1418688520
5      NA      NA      NA
6      NA 63064912 1764252

```

12.4 Conclusion

Getting the data just right to analyze can be a time-consuming part of our work flow, although it is often inescapable. In this chapter we examined combining multiple datasets into one and changing the orientation from column based (wide) to row based (long). We used `plyr`, `reshape2` and `data.table` along with base functions to accomplish this. This chapter combined with Chapter 11 covers most of the basics of data munging with an eye to both convenience and speed.

This page intentionally left blank

General Index

A

Addition
 matrices, 68
 order of operation, 36
 vectors, 44–45

Aggregation
 in `data.table` package, 135–138
 groups, 120–123

AICC, 320

Akaike Information Criterion (AIC), 255–257, 259–260

@aliases tag, 382

all.obs option, 196

Ampersands (&) in compound tests, 111

Analysis of variance (ANOVA)
 alternative to, 214–216
 cross-validation, 259–260
 model comparisons, 254
 overview, 207–210

And operator in compound tests, 111–112

Andersen–Gill analysis, 244–245

Angle brackets (<>)
 packages, 375
 regular expressions, 169

ANOVA. *See* Analysis of variance (ANOVA)

Ansari–Bradley test, 204

Appearance options, 21–22

Appending elements to lists, 68

apt-get mechanism, 2

Arguments
 C++ code, 385
 CSV files, 74
 functions, 49, 100–102
 ifelse, 110
 package documentation, 380

Arithmetic mean, 187

ARMA model, 315

Arrays, 71–72

Assigning variables, 36–37

Asterisks (*)
 Markdown, 368

 multiple regression, 228
 NAMESPACE file, 377
 vectors, 44

Attributes for `data.frame`, 54

Author information
 \LaTeX documents, 360
 packages, 375

@author tag, 382

Autocompleting code, 15–16

Autocorrelation, 318

Autoregressive (AR) moving averages, 315–322

Average linkage methods, 352, 355

Axes in nonlinear least squares model, 298

B

Back ticks (`) with functions, 49

Backslashes (\) in regular expressions, 166

Base graphics, 83–84
 boxplots, 85–86
 histograms, 84
 scatterplots, 84–85

Bayesian Information Criterion (BIC), 255–257, 259

Bayesian shrinkage, 290–294

Beamer mode in \LaTeX , 369

Beginning of lines in regular expressions, 167

Bell curve, 171

Bernoulli distribution, 176

Beta distribution, 185–186

BIC (Bayesian Information Criterion), 255–257, 259

Binary files, 77–79

Binomial distribution, 176–181, 185–186

Bioconductor, 373

BitBucket repositories, 25, 31

Books, 394

bootstrap, 262–265

Boxplots
 `ggplot2`, 91–94
 overview, 85–86

break statement, 115–116

Breakpoints for splines, 302

Building packages, 383–384

Byte-compilation for packages, 376

ByteCompile field, 376

C

C++ code, 384–383
 package compilation, 387–390
 sourceCpp function, 385–387

cache option for knitr chunks, 365

Calling functions, 49
 arguments, 100
 C++, 384
 conflicts, 33

Carets (^) in regular expressions, 167

Case sensitivity
 characters, 40
 package names, 384
 regular expressions, 162
 variable names, 38

Cauchy distribution, 185–186

Cauchy priors in Bayesian shrinkage, 293–294

Causation vs. correlation, 199

Censored data in survival analysis, 240–241

Centroid linkage methods, 352, 355

Change Install Location option, 9

character data, 40

Charts, 329

chartsnthings site, 393

Chi-Squared distribution, 185–186

Chunks
 \LaTeX program, 362–365
 Markdown, 368

Citations in \LaTeX documents, 366

Classification trees, 311

- Clusters, 337
 - hierarchical, 352–357
 - K-means algorithm, 337–345
 - PAM, 345–352
 - registering, 283
 - code
 - autocompleting, 15–16
 - C++, 384–390
 - indenting, 99
 - running in parallel, 282
 - Code Editing options, 21
 - Coefficient plots
 - Bayesian shrinkage, 292–294
 - Elastic Net, 289–290
 - logistic regression, 236
 - model comparisons, 253–254
 - multiple regression, 226–228, 230–231
 - Poisson regression, 237–240
 - residuals, 247, 249
 - VAR, 324–325
 - Collate field for packages, 375–376
 - Colons (:)
 - vectors, 44–45
 - Color
 - boxplots, 92
 - K-means algorithm, 339, 341
 - LaTeX documents, 362
 - line graphs, 96
 - PAM, 350–351
 - scatterplots, 88–90
 - Column index for arrays, 71
 - Columns
 - cbind and rbind, 141–142
 - data.frame, 53, 58
 - data.table, 131–133
 - matrices, 68–70
 - Comma separated files (CSVs), 73–74
 - Command line interface, 14–15
 - comment option, 365
 - Comments, 46
 - knitr chunks, 365
 - package documentation, 381
 - Community edition, 10–11
 - Comparing
 - models, 253–257
 - multiple groups, 207–210
 - multiple variables, 192
 - vectors, 46
 - Compilation in C++
 - code, 384
 - packages, 387–390
 - Complete linkage methods, 352, 355
 - complete.obs option, 196
 - Components, installing, 5
 - Compound tests, 111–112
 - Comprehensive R Archive Network (CRAN), 1, 29, 384
 - Concatenating strings, 155–156
 - Conferences, 393
 - Confidence intervals
 - ANOVA, 207–209, 215–216
 - bootstrap, 262, 264–265
 - Elastic Net, 277, 279
 - GAM, 310
 - multiple regression, 226
 - one-sample t-tests, 200–203
 - paired two-sample t-tests, 207
 - two-sample t-tests, 205–206
 - Control statements, 105
 - compound tests, 111–112
 - if and else, 105–108
 - ifelse, 109–111
 - switch, 108–109
 - Converting shapefile objects into data.frame, 349
 - Correlation and covariance, 191–200
 - Covariates in simple linear regression, 211
 - Cox proportional hazards model, 242–244
 - .cpp files, 386
 - CRAN (Comprehensive R Archive Network), 1, 29, 384
 - Create Project options, 16–17
 - Cross tables, 149
 - Cross-validation
 - Elastic Net, 276–277
 - overview, 257–262
 - CSVs (comma separated files), 73–74
 - Cubic splines, 302
 - Curly braces ({})
 - functions, 99
 - if and else, 106–107
 - regular expressions, 166
-
- D
- Data
 - censored, 240–241
 - missing. *See* Missing data
 - Data Analysis Using Regression and Multilevel/Hierarchical Models*, 50, 291, 394
 - data folder, 373–374
 - data.frames, 53–61
 - converting shapefile objects into, 349
 - ddply function, 124, 126
 - Elastic Net, 272
 - joins, 145
 - merging, 143–144
 - Data Gotham conference, 393
 - Data meetups, 391
 - Data munging, 117
 - Data reshaping, 141
 - cbind and rbind, 141–142
 - joins, 142–149
 - reshape2 package, 149–153
 - Data structures, 53
 - arrays, 71–72
 - data.frame, 53–61
 - lists, 61–68
 - matrices, 68–71
 - Data types, 38
 - C++ code, 387
 - character, 40
 - dates, 40–41
 - logical, 41–43
 - matrices, 68
 - numeric, 38–39
 - vectors, 43–48
 - Databases, reading from, 75–76
 - Dates, 40–41
 - LaTeX documents, 360
 - packages, 375
 - Decision trees, 310–312
 - \DeclareGraphicsExtensions, 360
 - Default arguments, 101–102
 - Degrees of freedom
 - ANOVA, 215
 - multiple regression, 225
 - splines, 300
 - t-tests, 201–202
 - Delimiters in CSV files, 74
 - Delta in model comparisons, 258
 - Dendrograms
 - ggplot2, 87–88
 - hierarchical clustering, 352
 - normal distribution, 172–173
 - Density plots, 87–88, 184, 207
 - Dependencies in packages, 30
 - Dependent variables in simple linear regression, 211
 - Depends field
 - C++ code, 386
 - packages, 375
 - Description field, 374–375
 - DESCRIPTION file, 374–377
 - @description tag, 382
 - Destination in installation, 4–5
 - @details tag, 382
 - dev option for knitr chunks, 365
 - Deviance in model comparisons, 256
 - Diffing process, 318–319
 - Dimensions in K-means algorithm, 339

direction argument, 265

Directories
 creating, 18
 installation, 4
 names, 18

Distance between clusters, 352

Distance metric for K-means
 algorithm, 337

Distributions. *See* Probability distributions

Division
 matrices, 68
 order of operation, 36
 vectors, 44–45

Documentation
 functions, 49
 packages, 380–383

\documentclass, 360

Documents as R resources, 394

Dollar signs (\$)

- data.frame, 56
- multiple regression, 225
- regular expressions, 167

%dopar% operator, 284

dot-dot-dot argument (...), 102

Downloading R, 1–2

DSN connections, 75

Dynamic Documents with R and knitr, 394

dzslides format, 369

E

echo option for knitr chunks, 365

EDA (Exploratory data analysis), 83, 199, 219

Elastic Net, 271–290

Elements of Statistical Learning: Data Mining, Inference, and Prediction, 394

End of lines in regular expressions, 167

engine option for knitr chunks, 365

Ensemble methods, 312

Environment, 13–14

- command line interface, 14–15
- RStudio. *See* RStudio overview

Equal to symbol (=)

- if and else, 105
- variable assignment, 36

Equality of matrices, 68

Esc key in command line

- commands, 15

eval option for knitr chunks, 365

everything option, 196

@examples tag, 382

Excel data, 74–75

Exclamation marks (!) in Markdown, 368

Expected value, 188

Experimental variables in simple linear regression, 211

Exploratory data analysis (EDA), 83, 199, 219

Exponential distribution, 185–186

Exponents, order of operation, 36

@export tag, 382

Expressions, regular, 161–169

Extra arguments, 102

Extracting

- data from Websites, 80–81
- text, 157–161

F

F distribution, 185–186

F-tests

- ANOVA, 215
- multiple regression, 225
- simple linear regression, 214–215
- two-sample, 204

faceted plots, 89–92

factor data type, 40

factors

- as.numeric with, 160
- Elastic Net, 273
- storing, 60
- vectors, 48

FALSE value

- with if and else, 105–108
- with logical operators, 41–43

fig.cap option, 365–366

fig.scap option, 365

fig.show option, 365

fill argument for histograms, 87

Fitted values against residuals plots, 249–251

folder structure, 373

for loops, 113–115

Forests, random, 312–313

formula interface

- aggregation, 120–123
- ANOVA, 208
- Elastic Net, 272
- logistic regression, 235–236
- multiple regression, 224, 226, 230
- scatterplots, 84–85
- simple linear regression, 213

Formulas for distributions, 185–186

Frontend field for packages, 374

Functions

- arguments, 100–102
- assigned to objects, 99
- C++, 384
- calling, 49, 100
- conflicts, 33
- do.call, 104
- documentation, 49
- package documentation, 380
- return values, 103

G

g++ compiler, 385

Gamma distribution, 185–186

Gamma linear model, 240

GAMs (generalized additive models), 304–310

Gap statistic in K-means algorithm, 343–344

Garbage collection, 38

GARCH (generalized autoregressive conditional heteroskedasticity) models, 327–336

Gaussian distribution, 171–176

gcc compiler, 385

General options for RStudio tools, 20–21

Generalized additive models (GAMs), 304–310

Generalized autoregressive conditional heteroskedasticity (GARCH) models, 327–336

Generalized linear models, 233

- logistic regression, 233–237
- miscellaneous, 240
- Poisson regression, 237–240

Geometric distribution, 185–186

Git

- integration with RStudio, 25–26
- selecting, 19

Git/SVN option, 25

GitHub repositories, 25

- for bugs, 392
- package installation from, 31, 383
- README files, 380

Graphics, 83

- base, 83–86
- ggplot2, 86–97

Greater than symbols (>)

- if and else, 105
- variable assignment, 37

Groups, 117

- aggregation, 120–123
- apply family, 117–120
- comparing, 207–210
- data.table package, 129–138
- plyr package, 124–129

H

Hadoop framework, 117
 Hartigan's Rule, 340–342
 Hash symbols (#)
 comments, 46
 Markdown, 368
 package documentation, 381
 pandoc, 369
 header command in pandoc, 369
 Heatmaps, 193
 Hello, World! program, 99–100
 Help pages in package documentation, 381
 Hierarchical clustering, 352–357
 Histograms, 84
 bootstrap, 264
 ggplot2, 87–88
 multiple regression, 219
 Poisson regression, 238
 residuals, 253
 Hotspot locations, 297–298
 HTML tables, extracting data from, 80–81
 Hypergeometric distribution, 185–186
 Hypothesis tests in t-tests, 201–203

I

IDEs (Integrated Development Environments), 13–14
 if else statements, 105–108
 Images in L^AT_EX documents, 360
 @import tag, 382
 Imports field for packages, 375
 include option for knitr chunks, 365
 Indenting code, 99
 Independent variables in simple linear regression, 211
 Indexes
 arrays, 71
 data.table, 129
 L^AT_EX documents, 360
 lists, 66
 Indicator variables
 data.frame, 60
 Elastic Net, 273, 289–290
 multiple regression, 225
 PAM, 345
 Inferences
 ensemble methods, 312
 multiple regression, 216
 @inheritParams tag, 382
 Innovation distribution, 330
 Input variables in simple linear regression, 211

inst folder, 373–374
 Install dependencies option, 30
 install.packages command, 31
 Install Packages option, 30
 installing packages, 29–32, 383–384
 installing R, 2
 on Linux, 10
 on Mac OS X, 8–10
 on Windows, 2–7
 integer type, 38–39
 Integers in regular expressions, 166
 Integrated Development Environments (IDEs), 13–14
 Intel Matrix Kernel Library, 10
 Interactivity, 13
 Intercepts
 multiple regression, 216
 simple linear regression, 212–213
 Interquartile Range (IQR), 85–86
Introduction to R, 394
 Inverse gaussian linear model, 240
 IQR (Interquartile Range), 85–86
 Italics in Markdown, 367
 Iteration with loops, 113
 controlling, 115–116
 for, 113–115
 while, 115

J

Joining strings, 155–156
 Joins, 142–143
 data.table, 149
 merge, 143–144
 plyr package, 144–149
 Joint Statistical Meetings, 393

K

k-fold cross-validation, 257–258
 K-means algorithm, 337–345
 K-medoids, 345–352
 key columns with join, 144
 keys for data.table package, 133–135
 knots for splines, 302

L

L1 penalty, 271
 L2 penalty, 271
 Lags in autoregressive moving average, 318–319
 lambda functions, 279–282, 285–289
 Language selection, 3

lasso in Elastic Net, 271, 276, 279, 282
 L^AT_EX program
 installing, 359
 knitr, 362–367
 overview, 360–362
 Leave-one-out cross-validation, 258
 Legends in scatterplots, 89
 Length
 characters, 40
 lists, 66–67
 vectors, 45–46
 Less than symbols (<)
 if and else, 105
 variable assignment, 36
 letters vector, 70
 LETTERS vector, 70
 Levels
 Elastic Net, 273
 factors, 48, 60
 LICENSE file, 380
 Licenses
 Mac, 8–9
 packages, 373–375
 SAS, 77
 Windows, 3
 Line breaks in Markdown, 367
 Line graphs, 94–96
 Linear models, 211
 generalized, 233–240
 multiple regression, 216–232
 simple linear regression, 211–216
 LinkingTo field, 386
 Links
 C++ libraries, 386
 hierarchical clustering, 352, 355
 linear models, 240
 Markdown, 368
 Linux
 C++ compilers, 385
 downloading R, 1–2
 installation on, 10
 Lists
 data.table package, 136–138
 joins, 145–149
 lapply and sapply, 118–119
 Markdown, 367
 overview, 61–68
 Loading
 packages, 32–33
 rdata files, 162
 log-likelihood in AIC model, 255
 Log-normal distribution, 185–186
 logical data type, 41–43
 Logical operators
 compound tests, 111–112
 vectors, 46

Logistic distribution, 185–186
 Logistic regression, 233–237
 Loops, 113
 controlling, 115–116
 for, 113–115
 while, 115

M

Mac
 C++ compilers, 385
 downloading R, 1
 installation on, 8–10
 Machine learning, 304
Machine Learning for Hackers, 394
 Machine Learning meetups, 391
 Maintainer field for packages, 375
 makeCluster function, 283
 \makeindex, 360
 Makevars file, 386–389
 Makevars.win file, 386–389
 man folder, 373–374
 MapReduce paradigm, 117
 Maps
 heatmaps, 193
 PAM, 350–351
 Markdown tool, 367–369
 Math, 35–36
 Matrices
 with apply, 117–118
 with cor, 192
 Elastic Net, 272
 overview, 68–71
 VAR, 324
 Matrix Kernel Library (MKL), 10
 .md files, 369–371
 Mean
 ANOVA, 209
 bootstrap, 262
 calculating, 187–188
 normal distribution, 171
 Poisson regression, 237–238
 t-tests, 203, 205
 various statistical distributions,
 185–186
 Mean squared error in
 cross-validation, 258
 Measured variables in simple linear
 regression, 211
 Meetups, 391–392
 Memory in 64-bit versions, 2
 Merging
 data.frame, 143–144
 data.table, 149
 Minitab format, 77

Minus signs (-) in variable assignment,
 36–37
 Missing data, 50
 apply, 118
 cor, 195–196
 cov, 199
 mean, 188
 NA, 50
 NULL, 51
 PAM, 346
 MKL (Matrix Kernel Library), 10
 Model diagnostics, 247
 bootstrap, 262–265
 comparing models, 253–257
 cross-validation, 257–262
 residuals, 247–253
 stepwise variable selection,
 265–269
 Moving average (MA) model, 315
 Moving averages, autoregressive,
 315–322
 Multicollinearity in Elastic Net, 273
 Multidimensional scaling in K-means
 algorithm, 339
 Multinomial distribution, 185–186
 Multinomial regression, 240
 Multiple group comparisons, 207–210
 Multiple imputation, 50
 Multiple regression, 216–232
 Multiple time series in VAR, 322–327
 Multiplication
 matrices, 69–71
 order of operation, 36
 vectors, 44–45
 Multivariate time series in VAR, 322

N

na.or.complete option, 196
 na.rm argument
 cor, 195–196
 mean, 188
 standard deviation, 189
 NA value
 with mean, 188
 overview, 50
 Name-value pairs for lists, 64
 Names
 arguments, 49, 100
 data.frame columns, 58
 directories, 18
 lists, 63–64
 packages, 384
 variables, 37–38
 vectors, 47
 names function for data.frame, 54–55

NAMESPACE file, 377–379
 Natural cubic splines, 302
 Negative binomial distribution,
 185–186
 Nested indexing of list elements, 66
 NEWS file, 379
 Nodes in decision trees, 311–312
 Noise
 autoregressive moving average,
 315
 VAR, 324
 Nonlinear models, 297
 decision trees, 310–312
 generalized additive model,
 304–310
 nonlinear least squares model,
 297–299
 random forests, 312–313
 splines, 300–304
 Nonparametric Ansari-Bradley test,
 204
 Normal distribution, 171–176
 Not equal symbols (!=) with if and
 else, 105
 nstart argument, 339
 Null hypotheses
 one-sample t-tests, 201–202
 paired two-sample t-tests, 207
 NULL value, 50–51
 Numbers in regular expressions,
 165–169
 numeric data, 38–39

O

Objects, functions assigned to, 99
 Octave format, 77
 1/mu^2 function, 240
 one-sample t-tests, 200–203
 Operations
 order, 36
 vectors, 44–48
 Or operators in compound tests,
 111–112
 Order of operations, 36
 Ordered factors, 48
 out.width option, 365
 Outcome variables in simple linear
 regression, 211
 Outliers in boxplots, 86
 Overdispersion in Poisson regression,
 238
 Overfitting, 312

P

p-values
 ANOVA, 208
 multiple regression, 225
 t-tests, 200–203

Package field in DESCRIPTION file, 374–377

Packages, 29, 373
 building, 33
 C++ code, 384–390
 checking and building, 383–384
 compiling, 387–390
 DESCRIPTION file, 374–377
 documentation, 380–383
 files overview, 373–374
 folder structure, 373
 installing, 29–32, 383–384
 loading, 32–33
 miscellaneous files, 379–380
 NAMESPACE file, 377–379
 options, 23
 submitting to CRAN, 384
 uninstalling, 32
 unloading, 33

Packages pane, 29–30

Paired two-sample t-tests, 206–207

pairwise.complete option, 197

PAM (Partitioning Around Medoids), 345–352

pandoc utility, 369–371

Pane Layout options, 21–22

Parallel computing, 282–284

@param tag, 381–382

Parentheses ()
 arguments, 100
 compound tests, 111
 expressions, 63
 functions, 99
 if and else, 105
 order of operation, 36
 regular expressions, 163

Partial autocorrelation, 318–319

Partitioning Around Medoids (PAM), 345–352

Passwords in installation, 9

Patterns, searching for, 161–169

PDF files, 362, 369

Percent symbol (%) in pandoc, 369

Periods (.)
 uses, 99
 variable names, 37

Plots
 coefficient. *See* Coefficient plots
 faceted, 89–92

Q–Q, 249, 252

residuals, 250–251

scatterplots. *See* Scatterplots

silhouette, 346–348

Plus signs (+) in regular expressions, 169

Poisson distribution, 182–184

Poisson regression, 237–240

POSIXct data type, 40

Pound symbols (#)
 comments, 46
 Markdown, 368
 package documentation, 381
 pandoc, 369

Prediction in GARCH models, 335

Predictive Analytics meetups, 391

Predictors
 decision trees, 310–311
 Elastic Net, 272
 generalized additive models, 304
 logistic regression, 233
 multiple regression, 216–217
 simple linear regression, 211, 213
 splines, 302–303

Priors, 290, 293–294

Probability distributions, 171
 binomial, 176–181
 miscellaneous, 185–186
 normal, 171–176
 Poisson, 182–184

Program Files\R directory, 4

Projects in RStudio, 16–19

prompt option for knitr chunks, 365

Q

Q–Q plots, 249, 252

Quantiles
 binomial distribution, 181
 multiple regression, 225
 normal distribution, 175–176
 summary function, 190

Quasibinomial linear model, 240

Quasipoisson family, 239

Question marks (?)
 with functions, 49
 regular expressions, 169

Quotes (") in CSV files, 74

R

R-Bloggers site, 393

R CMD commands, 383

R Enthusiasts site, 393

R folder, 373–374

R in Finance conference, 393

R *Inferno*, 394

R Productivity Environment (RPE), 26–27

Raise to power function, 45

Random numbers
 binomial distribution, 176
 normal distribution, 171–172

Random starts in K-means algorithm, 339

Rcmdr interface, 14

.Rd files, 380, 383

RData files
 creating, 77
 loading, 162

Readability of functions, 99

Reading data, 73
 binary files, 77–79
 CSVs, 73–74
 from databases, 75–76
 Excel, 74–75
 included with R, 79–80
 from statistical tools, 77

README files, 380

Real-life resources, 391
 books, 394
 conferences, 393
 documents, 394
 meetups, 391–392
 Stack Overflow, 392
 Twitter, 393
 Web sites, 393

Reference Classes system, 377

Registering clusters, 283

Regression
 generalized additive models, 304
 logistic, 233–237
 multiple, 216–232
 Poisson, 237–240
 simple linear, 211–216
 survival analysis, 240–245

Regression to the mean, 211

Regression trees, 310

Regular expressions, 161–169

Regularization and shrinkage, 271
 Bayesian shrinkage, 290–294
 Elastic Net, 271–290

Relationships
 correlation and covariance, 191–200
 multiple regression, 216–232
 simple linear regression, 211–216

Removing variables, 37–38

Repeating command line commands, 15

Reshaping data, 141
 cbind and rbind, 141–142
 joins, 142–149
 reshape2 package, 149–153

Residual standard error in least squares model, 298

Residual sum of squares (RSS), 254–255

Residuals, 247–253

Resources. *See* Real-life resources

Responses
 decision trees, 310
 logistic regression, 233
 multiple regression, 216–217, 219, 225
 Poisson regression, 237
 residuals, 247
 simple linear regression, 211–213

@return tag, 381–382

Return values in functions, 103

Revolution Analytics site, 393

Ridge in Elastic Net, 271, 279

.Rmd files, 369

.Rnw files, 362

Rows
 in arrays, 71
 bootstrap, 262
 cbind and rbind, 141–142
 data.frame, 53
 data.table, 131
 with mapply, 120
 matrices, 68–70

RPE (R Productivity Environment), 26–27

RSS (residual sum of squares), 254–255

RStudio overview, 15–16
 Git integration, 25–26
 projects, 16–19
 tools, 20–25

RTools, 385

Run as Administrator option, 3

Running code in parallel, 283

S

S3 system, 377

@S3method tag, 382

S4 system, 377

s5 slide show format, 369

SAS format, 77

Scatterplots, 84–85
 correlation, 192
 generalized additive models, 307
 ggplot2, 88–91

multiple regression, 220–224
 splines, 303

scope argument, 265

Scraping web data, 81

Seamless R and C++ Integration with Rcpp, 394

Searches, regular expressions for, 161–169

Secret weapon, 293

Sections in L^AT_EX documents, 361

@seealso tag, 382

Seeds for K-means algorithm, 338

Semicolons (;) for functions, 100

sep argument, 155

Shapefile objects, converting into data.frame, 349

Shapiro-Wilk normality test, 204

Shortcuts, keyboard, 15

Shrinkage
 Bayesian, 290–294
 Elastic Net, 271

Silhouette plots, 346–348

Simple linear regression
 ANOVA alternative, 214–216
 overview, 211–214

Single linkage methods, 352, 355

64-bit vs. 32-bit R, 2

Size
 binomial distributions, 176–179
 lists, 65
 sample, 187

Slashes (/) in C++ code, 385–386

Slide show formats, 369

slideous slide show format, 369

slidy format, 369, 371

Slope in simple linear regression, 212–213

Small multiples, 89

Smoothing functions in GAM, 304

Smoothing splines, 300–301

Software license, 3

Spelling options, 23–24

Splines, 300–304

Split-apply-combine method, 117, 124

SPSS format, 77

Square brackets ([])
 arrays, 71
 data.frame, 56, 58
 lists, 65
 Markdown, 368
 vectors, 47

Squared error loss in nonlinear least squares model, 297

src folder, 373–374, 387

Stack Overflow source, 392

Standard deviation
 missing data, 189
 normal distribution, 171
 simple linear regression, 213
 t-tests, 201–202, 205

Standard error
 Elastic Net, 279, 289
 least squares model, 298
 multiple regression, 225–226
 simple linear regression, 213–216
 t-tests, 202

start menu shortcuts, 6

startup options, 5

Stata format, 77

Stationarity, 318

Statistical graphics, 83
 base, 83–86
 ggplot2, 86–97

Statistical tools, reading data from, 77

Stepwise variable selection, 265–269

Strings, 155
 joining, 155–156
 regular expressions, 161–169
 sprintf, 156–157
 text extraction, 157–161

stringsAsFactors argument, 75

Submitting packages to CRAN, 384

Subtraction
 matrices, 68
 order of operation, 36
 vectors, 44–45

Suggests field in packages, 375–376

Summary statistics, 187–191

Survival analysis, 240–245

SVN repository, 17, 19, 25

switch statements, 108–109

Systat format, 77

T

t distribution
 functions and formulas, 185–186
 GARCH models, 330

t-statistic, 201–202, 225

t-tests, 200
 multiple regression, 225
 one-sample, 200–203
 paired two-sample, 206–207
 two-sample, 203–206

Tab key for autocompleting code, 15

Tables of contents in pandoc, 371

Tags for roxygen2, 381–382

Tensor products, 308

test folder, 374

Text

- extracting, 157–161
 - L^AT_EX documents, 362
 - regular expressions, 167–169
- Themes in `ggplot2`, 96–97
- 32-bit vs. 64-bit R, 2
- Tildes (~) in aggregation, 120
- Time series and autocorrelation, 315
 - autoregressive moving average, 315–322
 - GARCH models, 327–336
 - VAR, 322–327
- Title field, 374–375
- @title tag, 382
- Titles
 - help files, 381
 - L^AT_EX documents, 360
 - packages, 374–375
 - slides, 369
- Transposing matrices, 70
- Trees
 - decision, 310–312
 - hierarchical clustering, 354
- TRUE value
 - with if and else, 105–108
 - with logical operators, 41–43
- Twitter resource, 393
- Two-sample t-tests, 203–206
- Type field for packages, 374–375
- Types. *See* Data types

U

- Underscores (`_`)
 - Markdown, 367
 - variable names, 37
- Unequal length vectors, 46
- Uniform (Continuous) distribution, 185–186
- Uninstalling packages, 32
- Unloading packages, 33
- @useDynLib tag, 382
- useful package, 273, 341

- UseMethod command, 377
- useR! conference, 393
- User installation options, 9

V

- VAR (vector autoregressive) model, 322–327
- Variables, 36
 - assigning, 36–37
 - names, 37
 - relationships between, 211–216
 - removing, 37–38
 - stepwise selection, 265–269
- Variance, 189
 - ANOVA, 207–210
 - GARCH models, 327
 - Poisson regression, 238
 - t-tests, 203
 - various statistical distributions, 185–186
- Vector autoregressive (VAR) model, 322–327
- Vectorized arguments with `ifelse`, 110
- Vectors, 43–44
 - `data.frame`, 56
 - factors, 48
 - in for loops, 113–114
 - multiple regression, 217
 - multiplication, 44–45
 - operations, 44–48
 - paste, 155–156
 - `sprintf`, 157
- Version control, 19
- Version field for packages, 375
- version number, saving, 6–7
- Versions, 2
- Vertical lines (`|`) in compound tests, 111
- vim mode, 21
- Violins plots, 91–94
- Volatility in GARCH models, 330

W

- Weakly informative priors, 290
- Websites
 - extracting data from, 80–81
 - R resources, 393
- Weibull distribution, 185–186
- Welch two-sample t-tests, 203
- while loops, 115
- White noise
 - autoregressive moving average, 315
 - VAR, 324
- WiFi hotspot locations, 297–298
- Windows
 - C++ compilers, 385
 - downloading R, 1
 - installation on, 2–7
- Windows Live Writer, 15
- within-cluster dissimilarity, 343
- Wrapper functions, 386
- Writing R Extensions*, 394

X

- X-axes in nonlinear least squares model, 298
- Xcode, 385

Y

- Y-axes in nonlinear least squares model, 298
- y-intercepts
 - multiple regression, 216
 - simple linear regression, 212–213

Z

- Zero Intelligence Agents site, 393
- zypper mechanism, 2