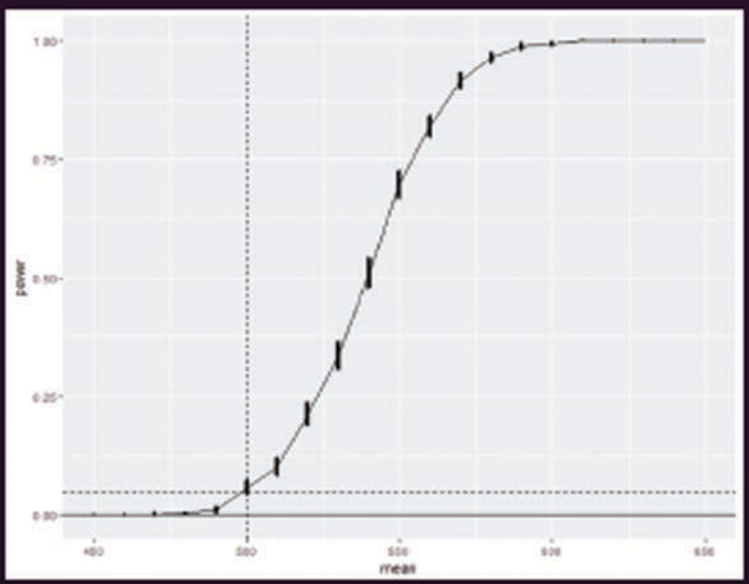
The R Series

Statistical Computing with R

Second Edition



Maria L. Rizzo

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Statistical Computing with R

Second Edition

Chapman & Hall/CRC
The R Series

Recently Published Titles

Using the R Commander: A Point-and-Click Interface for R
John Fox

Computational Actuarial Science with R
Arthur Charpentier

bookdown: Authoring Books and Technical Documents with R Markdown
Yihui Xie

Testing R Code
Richard Cotton

R Primer, Second Edition
Claus Thorn Ekstrøm

Flexible Regression and Smoothing: Using GAMLSS in R
Mikis D. Stasinopoulos, Robert A. Rigby, Gillian Z. Heller, Vlasios Voudouris, and Fernanda De Bastiani

The Essentials of Data Science: Knowledge Discovery Using R
Graham J. Williams

blogdown: Creating Websites with R Markdown
Yihui Xie, Alison Presmanes Hill, Amber Thomas

Handbook of Educational Measurement and Psychometrics Using R
Christopher D. Desjardins, Okan Bulut

Displaying Time Series, Spatial, and Space-Time Data with R, Second Edition
Oscar Perpinan Lamigueiro

Reproducible Finance with R
Jonathan K. Regenstein, Jr

R Markdown: The Definitive Guide
Yihui Xie, J.J. Allaire, Garrett Grolemund

Practical R for Mass Communication and Journalism
Sharon Machlis

Analyzing Baseball Data with R, Second Edition
Max Marchi, Jim Albert, Benjamin S. Baumer

Spatio-Temporal Statistics with R
Christopher K. Wikle, Andrew Zammit-Mangion, and Noel Cressie

Statistical Computing with R, Second Edition
Maria L. Rizzo

For more information about this series, please visit: https://www.crcpress.com/go/the-r-series

Statistical Computing with R

Second Edition

Maria L. Rizzo

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20190117

International Standard Book Number-13: 978-1-4665-5332-3 (Hardback)

Library of Congress Cataloging-in-Publication Data

Names: Rizzo, Maria L., author.
Title: Statistical computing with R / Maria L. Rizzo.
Description: Second edition. | Boca Raton : CRC Press, Taylor & Francis Group, 2019.
Identifiers: LCCN 2018049266| ISBN 9781466553323 (hardback) | ISBN 9781466553330 (ebook : alk. paper)
Subjects: LCSH: Mathematical statistics--Data processing. | Statistics--Data processing. | (Computer program language)
Classification: LCC QA276.45.R3 R59 2019 | DDC 519.50285/5133--dc23
LC record available at https://lccn.loc.gov/2018049266

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface to the Second Edition xi

Preface to the First Edition xiii

1 Introduction 1

1.1 Statistical Computing . . . 1
1.2 The R Environment . . . 3
1.3 Getting Started with R and RStudio . . . 5
1.4 Basic Syntax . . . 7
1.5 Using the R Online Help System . . . 9
1.6 Distributions and Statistical Tests . . . 11
1.7 Functions . . . 12
1.8 Arrays, Data Frames, and Lists . . . 13
1.9 Formula Specification . . . 20
1.10 Graphics . . . 20
1.11 Introduction to `ggplot` . . . 23
1.12 Workspace and Files . . . 26
1.12.1 The Working Directory . . . 28
1.12.2 Reading Data from External Files . . . 28
1.12.3 Importing/Exporting .csv Files . . . 31
1.13 Using Scripts . . . 32
1.14 Using Packages . . . 33
1.15 Using R Markdown and knitr . . . 33

2 Probability and Statistics Review 37

2.1 Random Variables and Probability . . . 37
2.2 Some Discrete Distributions . . . 42
2.3 Some Continuous Distributions . . . 45
2.4 Multivariate Normal Distribution . . . 49
2.5 Limit Theorems . . . 50
2.6 Statistics . . . 51
2.7 Bayes' Theorem and Bayesian Statistics . . . 55
2.8 Markov Chains . . . 57

3 Methods for Generating Random Variables 61

3.1 Introduction . . . 61
3.2 The Inverse Transform Method . . . 63
3.2.1 Inverse Transform Method, Continuous Case . . . 64
3.2.2 Inverse Transform Method, Discrete Case . . . 65
3.3 The Acceptance-Rejection Method . . . 69
3.4 Transformation Methods . . . 71
3.5 Sums and Mixtures . . . 75
3.6 Multivariate Distributions . . . 83
3.6.1 Multivariate Normal Distribution . . . 83
3.6.2 Mixtures of Multivariate Normals . . . 90
3.6.3 Wishart Distribution . . . 92
3.6.4 Uniform Distribution on the d-Sphere . . . 93
Exercises . . . 96

4 Generating Random Processes 99

4.1 Stochastic Processes . . . 99
4.1.1 Poisson Processes . . . 99
4.1.2 Renewal Processes . . . 104
4.1.3 Symmetric Random Walk . . . 105
4.2 Brownian Motion . . . 109
Exercises . . . 112

5 Visualization of Multivariate Data 115

5.1 Introduction . . . 115
5.2 Panel Displays . . . 115
5.3 Correlation Plots . . . 118
5.4 Surface Plots and 3D Scatter Plots . . . 120
5.4.1 Surface Plots . . . 121
5.4.2 Three-dimensional Scatterplot . . . 124
5.5 Contour Plots . . . 126
5.6 Other 2D Representations of Data . . . 129
5.6.1 Andrews Curves . . . 129
5.6.2 Parallel Coordinate Plots . . . 132
5.6.3 Segments, Stars, and Other Representations . . . 133
5.7 Principal Components Analysis . . . 135
5.8 Other Approaches to Data Visualization . . . 141
5.9 Additional Resources . . . 143
Exercises . . . 143

6 Monte Carlo Integration and Variance Reduction 147

6.1 Introduction . . . 147
6.2 Monte Carlo Integration . . . 147
6.2.1 Simple Monte Carlo Estimator . . . 148
6.2.2 Variance and Efficiency . . . 152
6.3 Variance Reduction . . . 154
6.4 Antithetic Variables . . . 155
6.5 Control Variates . . . 159
6.5.1 Antithetic Variate as Control Variate . . . 162
6.5.2 Several Control Variates . . . 163
6.5.3 Control Variates and Regression . . . 163
6.6 Importance Sampling . . . 168
6.7 Stratified Sampling . . . 173
6.8 Stratified Importance Sampling . . . 176
Exercises . . . 178
R Code . . . 181

7 Monte Carlo Methods in Inference 183

7.1 Introduction . . . 183
7.2 Monte Carlo Methods for Estimation . . . 184
7.2.1 Monte Carlo Estimation and Standard Error . . . 184
7.2.2 Estimation of MSE . . . 185
7.2.3 Estimating a Confidence Level . . . 188
7.3 Monte Carlo Methods for Hypothesis Tests . . . 192
7.3.1 Empirical Type I Error Rate . . . 193
7.3.2 Power of a Test . . . 197
7.3.3 Power Comparisons . . . 200
7.4 Application: "Count Five" Test for Equal Variance . . . 204
Exercises . . . 209

8 Bootstrap and Jackknife 213

8.1 The Bootstrap . . . 213
8.1.1 Bootstrap Estimation of Standard Error . . . 215
8.1.2 Bootstrap Estimation of Bias . . . 217
8.2 The Jackknife . . . 220
8.3 Bootstrap Confidence Intervals . . . 224
8.3.1 The Standard Normal Bootstrap Confidence Interval . . . 224
8.3.2 The Basic Bootstrap Confidence Interval . . . 225
8.3.3 The Percentile Bootstrap Confidence Interval . . . 226
8.3.4 The Bootstrap t Interval . . . 228
8.4 Better Bootstrap Confidence Intervals . . . 231
8.5 Application: Cross Validation . . . 235

Exercises . . . 239

9 Resampling Applications 243

9.1 Jackknife-after-Bootstrap . . . 243
9.2 Resampling for Regression Models . . . 246
9.2.1 Resampling Cases . . . 250
9.2.2 Resampling Errors (Model Based) . . . 254
9.3 Influence . . . 260
9.3.1 Empirical Influence Values for a Statistic . . . 260
9.3.2 Jackknife-after-Bootstrap Plots . . . 261
Exercises . . . 263

10 Permutation Tests 265

10.1 Introduction . . . 265
10.2 Tests for Equal Distributions . . . 269
10.3 Multivariate Tests for Equal Distributions . . . 272
10.3.1 Nearest Neighbor Tests . . . 273
10.3.2 Energy Test for Equal Distributions . . . 281
10.4 Application: Distance Correlation . . . 287
Exercises . . . 294

11 Markov Chain Monte Carlo Methods 297

11.1 Introduction . . . 297
11.1.1 Integration Problems in Bayesian Inference . . . 297
11.1.2 Markov Chain Monte Carlo Integration . . . 298
11.2 The Metropolis-Hastings Algorithm . . . 299
11.2.1 Metropolis-Hastings Sampler . . . 300
11.2.2 The Metropolis Sampler . . . 310
11.2.3 Random Walk Metropolis . . . 310
11.2.4 The Independence Sampler . . . 316
11.3 The Gibbs Sampler . . . 318
11.4 Monitoring Convergence . . . 322
11.4.1 Why Monitor Convergence . . . 322
11.4.2 Methods for Monitoring Convergence . . . 323
11.4.3 The Gelman-Rubin Method . . . 323
11.5 Application: Change Point Analysis . . . 327
Exercises . . . 333
R Code . . . 335

12 Probability Density Estimation **337**

12.1 Univariate Density Estimation . . . 337
12.1.1 Histograms . . . 338
12.1.2 Frequency Polygon Density Estimate . . . 345
12.1.3 The Averaged Shifted Histogram . . . 347
12.2 Kernel Density Estimation . . . 351
12.3 Bivariate and Multivariate Density Estimation . . . 361
12.3.1 Bivariate Frequency Polygon . . . 361
12.3.2 Bivariate ASH . . . 364
12.3.3 Multidimensional Kernel Methods . . . 366
12.4 Other Methods of Density Estimation . . . 369
Exercises . . . 370
R Code . . . 373

13 Introduction to Numerical Methods in R **375**

13.1 Introduction . . . 375
13.2 Root-finding in One Dimension . . . 383
13.3 Numerical Integration . . . 386
13.4 Maximum Likelihood Problems . . . 391
13.5 Application: Evaluating an Expected Value . . . 394
Exercises . . . 398

14 Optimization **401**

14.1 Introduction . . . 401
14.2 One-dimensional Optimization . . . 402
14.3 Maximum Likelihood Estimation with `mle` . . . 403
14.4 Two-dimensional Optimization . . . 405
14.5 The EM Algorithm . . . 409
14.6 Linear Programming – The Simplex Method . . . 411
14.7 Application: Game Theory . . . 413
Exercises . . . 417

15 Programming Topics **419**

15.1 Introduction . . . 419
15.2 Benchmarking: Comparing the Execution Time of Code . . . 419
15.2.1 Using the *microbenchmark* Package . . . 420
15.2.2 Using the *rbenchmark* Package . . . 423
15.3 Profiling . . . 425
15.4 Object Size, Attributes, and Equality . . . 427
15.4.1 Object Size . . . 427
15.4.2 Attributes of Objects . . . 428

15.4.3 Comparing Objects for Equality 429
15.5 Finding Source Code 430
15.5.1 Finding R Function Code 430
15.5.2 Methods 431
15.5.3 Methods and Functions in Packages 432
15.5.4 Compiled Code 433
15.6 Linking C/C++ Code Using *Rcpp* 434
15.7 Application: Baseball Data 438
Exercises 442

Notation **445**

Bibliography **447**

Index **469**

Preface to the Second Edition

In the years since the first edition of *Statistical Computing with R* appeared, many great enhancements in the R base, recommended packages, and thousands of contributed packages have developed, as well as important new tools for all aspects of writing and maintaining one's work in R. Indeed, on August 1, 2018, the CRAN package repository listed 12,860 available packages. Some of the major highlights are summarized below.

- RStudio: This excellent free integrated development environment (IDE) provides convenient editing of code, viewing of output and results, file management, package management, project management, and much more. It has become the most popular IDE for R users of all levels, for good reason. Its first public release was February 28, 2011 (0.92.23). It is available at `www.rstudio.org`. Chapter 1 "Introduction" has been revised accordingly.

- ggplot: ggplot and its successor ggplot2 [313] (ggplot2 release 0.7 on October 5, 2008: ggplot2-0.7) offers an alternate system of graphics that has had a great impact on much of the graphics content in R analyses. An introduction to ggplot is now included in Chapter 1 and some of the plots in the second edition are generated with ggplot or the ggplot code is provided.

- knitr: The introduction of the knitr package [323, 324] has greatly facilitated reproducible research with R. Reproducible research is ideally generated by a type of report which integrates the expository content, data input, data analysis, output and graphics seamlessly into one document, such that the entire project is reproducible through executing this document. An early form of this type of report was provided by the LaTeX Sweave package. Now the knitr package for R provides an engine for dynamic report generation within the RStudio IDE. It supports Sweave as well as R Markdown, which produces html or Word format output without LaTeX. It is so easy to use that first-year undergraduates can produce beautiful reports to submit as assignments in all types of courses. My first assignment is usually to have students create an html report implementing a few of the examples in Chapter 1. Subsequently, all assignments are submitted in that format. Now Chapter 1 has a new Exercises section and that is the final exercise.

- tidyverse: `tidyverse.org` is a great collection of some almost essential utilities. It includes packages like dplyr [317] which makes reading and manipulating large data frames much faster, stringr [314] and lubridate [131] for easier operations with strings and dates, and the graphics package ggplot2 [313], to name a few. For readers who prefer a minimal installation, only *ggplot2* and *knitr* (and the packages required by them) will be needed to get started. Other packages will be needed for functions and/or data in later chapters, but most of tidyverse would not be used later in the book. However, most readers plan to use R for statistics apart from this book and may find it convenient to install all of tidyverse in one step.

- Rcpp: The Rcpp [82, 83] package makes it relatively straightforward to integrate C or C++ code to an R script or source a C++ file from an R script. It can all be managed without leaving the RStudio IDE.

The second edition includes new chapters and sections, some additional exercises, examples, applications, and some material has been re-organized. A new section on Principal Components Analysis has been added to the chapter "Visualization of Multivariate Data." The chapter "Methods for Generating Random Variables" has been split into two chapters, creating a new chapter "Generating Random Processes." A new chapter "Resampling Applications" follows "Bootstrap and Jackknife" with an expanded section on jackknife-after-bootstrap and some applications of resampling for linear models. The final chapter of the first edition, "Numerical Methods in R" has been split into "Introduction to Numerical Methods in R" and "Optimization." Finally, a new chapter "Programming Topics" covers topics for more experienced users such as benchmarking, profiling, object size, finding source code, and a very brief introduction to the Rcpp package to extend R with C++.

The extended example at the end of Chapter 15 replaces Appendix B of the first edition, which covered methods for extracting and manipulating data. The application applies basic methods as well as alternate methods using *dplyr* package. The two approaches are compared and also benchmarked. Appendix B of the first edition will remain available as an online supplement.

The second edition of *Statistical Computing with R* will be accompanied by some online supplements available on GitHub at `github.com/mariarizzo/SCR2e`. The R code for all of the examples in the text will be available as well as a few tutorials or extended examples on selected topics.

Maria L. Rizzo, Professor
Department of Mathematics and Statistics
Bowling Green State University

Preface to the First Edition

This book is an introduction to statistical computing and computational statistics. Computational statistics is a rapidly expanding area in statistical research and applications. It includes computationally intensive methods in statistics, such as Monte Carlo methods, bootstrap, MCMC, density estimation, nonparametric regression, classification and clustering, and visualization of multivariate data. Gentle [118] and Wegman [308] describe *computational statistics* as computationally intensive methods in statistics. *Statistical computing*, at least traditionally, focused on numerical algorithms for statistics (see e.g. Thisted [284]). Generally a book has only one of these terms in the title; for example, Givens and Hoeting's "*Computational Statistics*" [128] includes classical statistical computing topics in optimization, numerical integration, density estimation and smoothing, as well as the Monte Carlo and MCMC methods of computational statistics. We chose the title *"Statistical Computing with R"* for this book, which is both computational statistics and statistical computing, and perhaps emphasizes Monte Carlo and resampling methods more than the title would suggest.

R is a statistical computing environment based on the S language. The software is free under the terms of the Free Software Foundation's GNU General Public License. It is available for a wide variety of platforms including among others Linux, Windows, and MacOS. See `http://www.r-project.org/` for a description. All examples in the text are implemented in R.

This book is designed for graduate students or advanced undergraduates with preparation in calculus, linear algebra, probability and mathematical statistics. The text will be suitable for an introductory course in computational statistics, and may also be used for independent study. In addition, because of the computational nature of the material, this book serves as an excellent tutorial on the R language, providing examples that illustrate programming concepts in the context of practical computational problems. The text does not assume previous expertise in any particular programming language.

The presentation will focus on implementation rather than theory, but the connection to the mathematical ideas and theoretical foundations will be made clear. The first chapter provides an overview of computational statistics and a brief introduction to the R statistical computing environment. The second chapter is a summary and review of some basic concepts in probability and classical statistical inference. Each of the remaining chapters covers a topic in computational statistics.

The selection of topics includes the traditional core material of computational statistics: simulating random variables from probability distributions, Monte Carlo integration and variance reduction methods, Monte Carlo and MCMC methods, bootstrap and jackknife, density estimation, and visualization of multivariate data. Although R includes random generators for the commonly used probability distributions, there is instructive value in studying the algorithms for generating them. Research problems often involve distributions that are non-standard, generalized, or not implemented. Methods for generating mixtures and multivariate data are also covered. The text concludes with a chapter on numerical methods in R.

A large number of examples and exercises are included. All examples are fully implemented in the R statistical computing environment, and the R code for examples in the book can be downloaded from the author's web site at `personal.bgsu.edu/~mrizzo`. In an effort to keep the material self-contained, most examples and exercises use datasets available in the R distribution (base plus recommended packages), or simulated data. Some functions and datasets in contributed packages available on CRAN are used, which can be installed by functions provided in R.

Books in print have a long lifetime, while software is constantly evolving. By the time this book is in a reader's hands, one or more newer versions of R will have been released. Every effort has been made to check the code samples under the current version of R; comments, suggestions, and corrections are always welcome.

Acknowledgements

This book was inspired at least in part by the excellent statistical computing package R, and the author would like to acknowledge the team of developers for continuing to support and improve this software.

I would like to thank several reviewers who made invaluable suggestions and comments, especially Jim Albert, Hua Fang, Herb McGrath, Xiaoping Shen, and Gábor Székely. I would also like to acknowledge the contribution of my students who used a preliminary draft of the text at Ohio University and provided much helpful feedback, with special thanks to Roxana Hritcu, Nihar Shah, and Jinfei Zhang. Editor Bob Stern, Project Editor Marsha Hecht, and Project Coordinator Amber Donley of Taylor & Francis / CRC Press have been very helpful throughout the entire project. Finally, I would like to thank my family for their constant support and encouragement.

Maria L. Rizzo
Department of Mathematics and Statistics
Bowling Green State University

Chapter 1

Introduction

1.1 Statistical Computing

Computational statistics and statistical computing are two areas within statistics that may be broadly described as computational, graphical, and numerical approaches to solving statistical problems. *Statistical computing* traditionally has more emphasis on numerical methods and algorithms, such as optimization and random number generation, while *computational statistics* may encompass such topics as exploratory data analysis, Monte Carlo methods, and data partitioning, etc. However, most researchers who apply computationally intensive methods in statistics use both computational statistics and statistical computing methods; there is much overlap and the terms are used differently in different contexts and disciplines. Gentle [118] and Givens and Hoeting [129] use "computational statistics" to encompass all the relevant topics that should be covered in a modern introductory text, so that "statistical computing" is somewhat absorbed under this more broad definition of computational statistics. On the other hand, journals and professional organizations seem to use both terms to cover similar areas.

This book encompasses parts of both of these subjects, because a first course in computational methods for statistics necessarily includes both. Some examples of topics covered are described below.

Monte Carlo methods refer to a diverse collection of methods in statistical inference and numerical analysis where simulation is used. Many statistical problems can be approached through some form of Monte Carlo integration. In parametric bootstrap, samples are generated from a given probability distribution to compute probabilities, gain information about sampling distributions of statistics such as bias and standard error, to assess the performance of procedures in statistical inference, and to compare the performance of competing methods for the same problem. Resampling methods such as the ordinary bootstrap and jackknife are nonparametric methods that can be applied when the distribution of the random variable or a method to simulate it directly is unavailable. The need for Monte Carlo analysis also arises because in many problems, an asymptotic approximation is unsatisfactory or intractable. The convergence to the limit distribution may be too slow, or we require results for finite samples; or the asymptotic distribution has unknown parameters.

Monte Carlo methods are covered in Chapters 6–11. The first tool needed in a simulation is a method for generating psuedo-random samples; these methods are covered in Chapters 3 and 4.

Markov Chain Monte Carlo (MCMC) methods are based on an algorithm to sample from a specified target probability distribution that is the stationary distribution of a Markov chain. These methods are widely applied for problems arising in Bayesian analysis, and in such diverse fields as computational physics and computational finance. Markov Chain Monte Carlo methods are covered in Chapter 11.

Several special topics also deserve an introduction in a survey of computationally intensive methods. Density estimation (Chapter 12) provides a nonparametric estimate of a density, which has many applications in addition to estimation, ranging from exploratory data analysis to cluster analysis. Computational methods are essential for the visualization of multivariate data and reduction of dimensionality. The increasing interest in massive and streaming data sets, and high dimensional data arising in applications of biology and engineering, for example, demand improved and new computational approaches for multivariate analysis and visualization. Chapter 5 is an introduction to methods for visualization of multivariate data. A review of selected topics in numerical methods such as root finding and numerical integration is presented in Chapter 13. An introduction to optimization using R is covered in Chapter 14.

A final chapter of optional material specific to R programming should be accessible to readers after covering Chapter 3. Programming topics such as benchmarking, efficiency and code profiling are covered in Chapter 15. Several years ago with the release of *Rcpp* [82, 83], writing R extensions in compiled libraries became much simpler so that most experienced R users with a modest amount of background in C++ can easily integrate compiled C++ functions with R code. Some simple examples are illustrated in the final chapter of the book for those users who are interested.

Many references can be recommended for further reading on these topics. Efron and Hastie [89] provide an up-to-date review of how modern statistics has evolved in the computer age. Gentle [118, 119] and the volume edited by Gentle, et al. [120] have thorough coverage of topics in computational statistics. A survey of methods in statistical computing is covered in Kundu and Basu [170]. Givens and Hoeting [129] is a recent graduate text on computational statistics and statistical computing. Hardle et al. [139] is an introductory text with examples in R. Martinez and Martinez [197] is an accessible introduction to computational statistics, with numerous examples in MATLAB®. Books that primarily cover Monte Carlo methods or resampling methods include Davison and Hinkley [68], Efron and Tibshirani [91], Hjorth [149], Liu [186], Chernick [50] and Robert and Casella [240]. Statistical learning is a closely related topic that applies computational methods to solve a wide range of problems in modern statistics; see Hastie et al. [143] and James et al. [157]. On density estimation see Scott [264] and Silverman [268]. A good resource

for applied linear models in R and other extensions such as nonparametric regression and smoothing is Faraway [95]. Albert [5] and McElreath [199] cover Bayesian computational methods with examples in R. For statistical applications of numerical analysis see Lange [176] or Monahan [210].

Although this book aims to be complete for novice R users to get started, it is not intended as a full-length text about using R for statistics or data science. Some R users may also be interested in supplementary resources designed for learning to use R. There is a long list of introductory books and materials of this type. *R by Example* [7] may appeal to users who enjoy learning from detailed, fully implemented examples. Verzani [295], Dalgaard [67] or Wickham and Grolemund [318] are on a similar level. For graphics in R, see Chang's *R Graphics Cookbook* [47] and refer to both Chang [47] and Wickham [313] for *ggplot2*.

For technical reference on programming in R, several excellent references are available in addition to the R manuals [227, 229, 294]. For advanced programming topics see Eddelbuettel [82], Gillespie and Lovelace [127], and Wickham [312], and their respective websites.

There are now many excellent online resources available, in addition to the online R and RStudio documentation, such as galleries of code and graphics, online books, tutorials and blogs. See the references in the individual chapters for some of these. The *R-bloggers* website is worth visiting; it currently combines blog posts from some 750 bloggers at `https://www.r-bloggers.com/`.

1.2 The R Environment

The R environment is a suite of software and programming language based on S, for data analysis and visualization. "What is R" is one of the frequently asked questions included in the online documentation for R. Here is an excerpt from the R FAQ [226]:

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

R is based on the S language. Some details about differences between R and S are given in the R FAQ [151]. Venables and Ripley [293] is a good resource for applied statistics with S, Splus, and R. Other references on the S language include [27, 42, 45, 292].

The home page of the R project is `http://www.r-project.org/`, and the current R distribution and documentation are available on the Comprehensive R Archive Network (CRAN) at `http://cran.R-project.org/`. The R dis-

tribution includes the base and recommended packages with documentation. A help system and several reference manuals are installed with the program.

Readers who have not already done so should proceed to download and install the most recent version of R corresponding to their operating system. Installation is easy; after downloading the setup file from `http://cran.R-project.org/` and running it, most users will simply accept the default options as prompted by the installation wizard.

Most R users currently use an integrated development environment or IDE to interact with the R system, edit source files, and view output. Although a type of IDE (the R GUI) is included with the R distribution, it is no longer widely used. Perhaps the most popular IDE for R currently is RStudio. In this second edition of the book, RStudio is treated as the default IDE as it is widely used, free to download the noncommercial version, and packed with convenient features. Users can install a free version of RStudio from `https://www.rstudio.com/`. RStudio has recently released RStudio Cloud, currently in alpha. For more information about the cloud option, consult the websites `https://rstudio.cloud/` and the community page at `https://community.rstudio.com/c/rstudio-cloud`. Other IDEs are available, of course, and any of them can easily be used with this book.

Programming is discussed as needed in the chapters that follow. In this text, new functions or programming methods are explained in remarks called "R notes" as they arise. Some "R notes" also address certain aspects of the R system or devices. Readers are always encouraged to consult the R help system and manuals [151, 226, 294]. For platform specific details about installation and interacting with the graphical user interface the best resources are the R manual [228] and current information at `www.r-project.org`.

Although RStudio provides many user-friendly features and powerful tools, R is a stand-alone program that could be run in batch mode if required for certain projects. R scripts can execute on a supercomputer and there are extensions to enable high performance computing. Other extensions like *rstan* provide an interface to a powerful scripting language and sampling engine *Stan* for Bayesian analysis. Refer to the CRAN Task Views "High Performance Computing" and "Bayesian" for more details. CRAN Task Views are a good resource to find what is available on CRAN for a wide range of statistical and machine learning applications. See `https://cloud.r-project.org/web/views/`.

In the remainder of this chapter, we cover some basic information aimed to help a new user get started with R. Topics include the recommended RStudio integrated development environment, basic syntax, using the online help, data, files, scripts, and packages. Vectors, matrices, lists and data frames are introduced with examples, and there is an overview of basic graphics functions.

1.3 Getting Started with R and RStudio

RStudio provides a convenient user interface that makes R much easier to use for beginners and advanced users. It is open source and available to download and install from the RStudio website at `https://www.rstudio.com/`. It combines a code editor with syntax highlighting, plot, environment, and console windows. An integrated help system and other important utilities are provided. RStudio has extensive support for generating reports using R Markdown with the *knitr* package [324], and package development without leaving the RStudio environment. A screen shot of an RStudio session is shown in Figure 1.1.

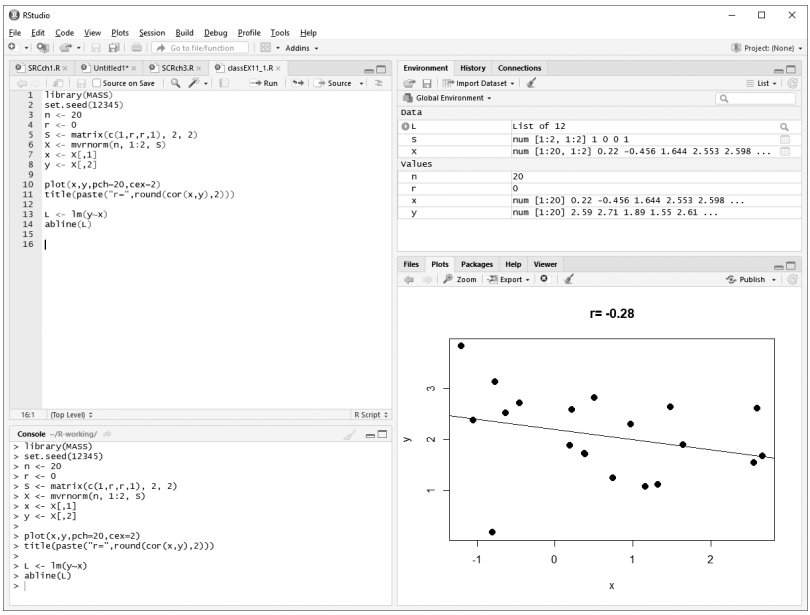


FIGURE 1.1: RStudio screen shot.

In the screen shot of RStudio, Figure 1.1, an R script is open in the code editor window (upper left), and it has been run interactively so that commands and results appear in the Console window (lower left) and Plot window (lower right). In the upper right the Environment window is visible, showing the names and values of user defined objects in the environment. To try a similar example, open the R code file for this chapter, use the mouse to select the first several lines, and click "Run" from the toolbar in the code editor window. Alternately click "Source" to run all code in the script.

Commands can also be typed at the prompt in the R Console window. For

example, we can evaluate the standard normal density $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ at $x = 2$ by typing the formula or (more conveniently) the `dnorm` function:

```
> 1/sqrt(2*pi) * exp(-2)
[1] 0.05399097
> dnorm(2)
[1] 0.05399097
```

In the example above, the command prompt is `>`. The `[1]` indicates that the result displayed is the first element of a vector.

A command can be continued on the next line. The prompt symbol changes whenever the command on the previous line is not complete. In the example below, the plot command is continued on the second line, as indicated by the prompt symbol changing to `+`.

```
> plot(cars, xlab="Speed", ylab="Distance to Stop",
+ main="Stopping Distance for Cars in 1920")
```

Whenever a statement or expression is not complete at the end of a line, the parser automatically continues it on the next line. No special symbol is needed to end a line. (A semicolon can be used to separate statements on a single line, although this tends to make code harder to read.) A group of statements can be gathered into a single (compound) expression by enclosing them in curly braces `{ }`.

To cancel a command, a partial command, or a running script, use Ctrl-C, or in Windows press the escape key (Esc). If the RStudio console window has a red square button, an error has occurred and one can debug or click the red square to stop.

To exit the RStudio IDE, simply close the main window. The program usually prompts with the question "Save workspace data to /.Rdata?" Click yes to save the workspace, which includes user defined objects and remembers any open files, or click "No" to exit without saving.

R Note 1.1 Why are some results not seen in the console?

If R code is submitted interactively in RStudio using "Run", the R code and result are echoed to the console, but when a file is sourced using the "Source" button or `source` function, statements and results are not echoed to the console. For example, evaluating the expression `dnorm(2)` interactively by typing in the console window or using "Run" echoes the command and prints the result to the console. However, if that expression is part of an R script, when the script is sourced using "Source," the result is not printed unless we explicitly print it, e.g., `print(dnorm(2))`. In RStudio, the "Source" button has a drop-down menu with an optional "Source with echo" in case this is an issue.

R Note 1.2

The RStudio Help menu includes RStudio Docs, which links to an online support page. A good article to look over on this page is 'Using the RStudio IDE' which covers many basic and less obvious features of RStudio. A very useful feature for editing within RStudio is that one or more Source panes can be detached and moved to the desktop. Simply click on the tab, drag and drop. Also see Keyboard Shortcuts Help for shortcuts to comment/uncomment lines, reformat code, and many other actions; a very handy shortcut Ctrl+Enter will run the selected line(s) of code or the line containing the cursor. To exit the shortcut help, press Esc.

1.4 Basic Syntax

The usual assignment operator is `<-`. For example, `x <- sqrt(2 * pi)` assigns the value of $\sqrt{2\pi}$ to the symbol `x`.

R Note 1.3 The assignment operator

In many situations the two assignment operators `<-` and `=` can be used interchangeably. It is a good practice to use `<-` for assignment because there is a technical difference between the two operators. The R documentation on assignment operators states that "The operators <- and = assign into the environment in which they are evaluated. The operator <- can be used anywhere, whereas the operator = is only allowed at the top level (e.g., in the complete expression typed at the command prompt) or as one of the subexpressions in a braced list of expressions." Throughout this book we will use `<-` for assignment and reserve `=` for arguments to functions.

Commands entered at the command prompt in the R console are automatically echoed to the console, but assignment operations are silent. Some objects have print methods so that the output displayed is not necessarily the entire object, but a summarized report. Compare the effect of these commands. The first command displays a sequence (0.0 0.5 1.0 1.5 2.0 2.5 3.0), but does not store it. The second command stores the sequence in x, but does not display it.

```
seq(0, 3, 0.5)
x <- seq(0, 3, 0.5)
```

Syntax

Below are some help topics on R operators and syntax. The `?` invokes the help system for the indicated keyword.

```
?Syntax
?Arithmetic
?Comparison  #relational operators
?Extract     #operators on vectors and arrays
?Control     #control flow
?Logic       #logical operators
```

Symbols or labels for functions and variables are case-sensitive and can include letters, digits, and periods. Symbols cannot contain the underscore character and cannot start with a digit. Many symbols are already defined by the R base or recommended packages. To check if a symbol is already defined, type the symbol at the prompt. The symbols `q`, `t`, `I`, `T`, and `F`, for example, are used by R. Note that whenever a package is loaded, other symbols may now be defined by the package.

```
> T
[1] TRUE
> t
function (x) UseMethod("t") <environment: namespace:base>
> g
Error: Object "g" not found
```

Here we see that both `T` and `t` are already defined, but `g` is not yet defined by R or by the user. Nothing prevents a user from assigning a new value to predefined symbols such as `t` or `T`, but it is a bad programming practice in general and can lead to unexpected results and programming errors.

Most new R users have some experience with other programming environments and languages such as C, MATLAB, or SAS. Some operations and features are common to all these languages. A brief list summarizing R syntax for some of these common elements is shown in Table 1.1. For more details see the help topic `Syntax`. Some of the functions common to most development environments are listed in Table 1.2.

Most arithmetic operations are vectorized. For example, `x^2` will square each of the elements of the vector `x`, or each entry of the matrix `x` if `x` is a matrix. Similarly, `x*y` will multiply each of the elements of the vector `x` times the corresponding element of `y` (generating a warning if the vectors are not the same length). Operators for matrices are described in Table 1.3.

TABLE 1.1: R Syntax and Commonly Used Operators

Description	R symbol	Example
Comment	`#`	`#this is a comment`
Assignment	`<-`	`x <- log2(2)`
Concatenation operator	`c`	`c(3,2,2)`
Elementwise multiplication	`*`	`a * b`
Exponentiation	`^`	`2^1.5`
x mod y	`x %% y`	`25 %% 3`
Integer division	`%/%`	`25 %/% 3`
Sequence from a to b by h	`seq`	`seq(a,b,h)`
Sequence operator	`:`	`0:20`

TABLE 1.2: Commonly Used Functions

Description	R symbol
Square root	`sqrt`
$\lfloor x \rfloor$, $\lceil x \rceil$	`floor`, `ceiling`
Natural logarithm	`log`
Exponential function e^x	`exp`
Factorial	`factorial`
Random Uniform numbers	`runif`
Random Normal numbers	`rnorm`
Normal distribution	`pnorm`, `dnorm`, `qnorm`
Rank, sort	`rank`, `sort`
Variance, covariance	`var`, `cov`
Std. dev., correlation	`sd`, `cor`
Frequency tables	`table`
Missing values	`NA`, `is.na`

1.5 Using the R Online Help System

RStudio includes a Help tab with a search box for searching by keyword. Help topics can also be searched from the command prompt. For documentation on a topic, type `?topic` or `help(topic)` where "topic" is the name of the topic for which you need help. For example, `?seq` will bring up documentation for the sequence function. In some cases, it may be necessary to surround the topic with quotation marks.

```
> ?seq   #display help for sequence function
> ?%%
Error: syntax error, unexpected SPECIAL in " ?%%"
```

The second version (below) produces the help topic.

```
> ?"%%"
```

TABLE 1.3: R Syntax and Functions for Vectors and Matrices

Description	R symbol	Example
Zero vector	`numeric(n)`	`x <- numeric(n)`
	`integer(n)`	`x <- integer(n)`
	`rep(0,n)`	`x <- rep(0,n)`
Zero matrix	`matrix(0,n,m)`	`x <- matrix(0,n,m)`
i^{th} element of vector a	`a[i]`	`a[i] <- 0`
j^{th} column of a matrix A	`A[,j]`	`sum(A[,j])`
ij^{th} entry of matrix A	`A[i,j]`	`x <- A[i,j]`
Matrix multiplication	`%*%`	`a %*% b`
Elementwise multiplication	`*`	`a * b`
Matrix transpose	`t`	`t(A)`
Matrix inverse	`solve`	`solve(A)`
Diagonal	`diag`	`diag(A)`

In RStudio, "R Help" in the Help menu displays Help in an integrated browser window, with hyperlinks. Alternately the function `help.start()` entered at the command prompt will display a summary of topics in html format with links.

Another way to search for help on a topic is `help.search()`. This and the search engine in Html help may help locate several relevant topics. For example, if we are searching for a method to compute a permutation,

```
help.search("permutation")
```

produces two results: `order` and `sample`. We can then consult the help topics for `order` and `sample`. The help topic for `sample` shows that x is sampled without replacement (a permutation of the elements of vector x) by:

```
sample(x)          #permutation of all elements of x
sample(x, size=k)  #permutation of k elements of x
```

(If the goal was to *count* permutations, and evaluate $\frac{n!}{(n-k)!}$, we want `?Special`, a list of special functions including `factorial` and `gamma`.)

Many help files end with executable examples. The examples can be copied and pasted at the command line. To run all the examples associated with `topic`, use `example(topic)`. See for example the interesting set of examples for `density`. To run all the examples for `density`, type `example(density)`. To see one example, open the help page, copy the lines and paste them at the command prompt.

```
help(density)
# copy and paste the lines below from the help page

# The Old Faithful geyser data
d <- density(faithful$eruptions, bw = "sj")
d
plot(d)
```

A list of available data sets in the base and loaded packages is displayed by `data()`, and documentation on a loaded data set is displayed by the associated help topic For example, `help("faithful")` displays the Old Faithful geyser data help topic. If a package is installed but not yet loaded, specify the name of the package. For example, `help("geyser", package = MASS)` displays help for the dataset `geyser` without loading the *MASS* package [293]. Alternately, `MASS::geyser` will access the `geyser` data from the *MASS* package. For example, to get the `summary` of this data:

```
> summary(MASS::geyser)

      waiting          duration
 Min.   : 43.00   Min.   :0.8333
 1st Qu.: 59.00   1st Qu.:2.0000
 Median : 76.00   Median :4.0000
 Mean   : 72.31   Mean   :3.4608
 3rd Qu.: 83.00   3rd Qu.:4.3833
 Max.   :108.00   Max.   :5.4500
```

1.6 Distributions and Statistical Tests

There are dozens of probability distributions and statistical tests implemented in the R *stats* package, which is automatically available when using R. To use the integrated help system to search for a list of what is available, search for the keyword "Distributions." This search should find a manual page that lists all of the available probability distributions included in *stats* when R is installed. Other distribution functions may be available in external packages.

To search for statistical tests implemented in R *stats*, it is easiest to use the wildcard type of search `help.search("keyword", package="stats")`. This restricts the search to the *stats* package so that we only see the results in that package. It is helpful to know that test functions in R are named in this pattern: "name.test". For example, Pearson's chisquared test function is `chisq.test`. A wildcard search ending in ".test" should find all of the test functions.

For example, try the following searches.

```
help.search("distribution", package="stats")
help.search(".test", package="stats")
```

The above search for "distribution" displays a list of links to help pages for statistical distributions in *stats*, along with a few other distribution-related

functions such as the empirical distribution function (`ecdf`). There is also a link to the "Distributions" manual page.

The search for ".test" displays a list of links to help pages for over 30 statistical tests implemented in R *stats*. The list includes `t.test` (Student's T test), `cor.test` (correlation test), `prop.test` (tests for proportions), and many other commonly used tests. See Example 1.7 for an application of the Wilcoxon rank sum test.

1.7 Functions

The syntax for a function definition is

```
function( arglist ) expr
return(value)
```

Many examples of functions are documented in the chapter "Writing your own functions" of the manual [294].

Example 1.1. Here is a simple example of a user-defined R function that "rolls" n fair dice and returns the sum.

```
sumdice <- function(n) {
    k <- sample(1:6, size=n, replace=TRUE)
    return(sum(k))
}
```

The function definition can be entered by several methods.

1. Typing the lines at the prompt, if the definition is short.
2. Copy from an editor and paste at the command prompt.
3. Save the function in a script file and source the file.

Note that the IDE provides an editor and toolbar for submitting code. Once the user-defined function is entered in the workspace, it can be used like other R functions.

```
#to print the result at the console
> sumdice(2)
[1] 9

#to store the result rather than print it
a <- sumdice(100)

#we expect the mean for 100 dice to be close to 3.5
> a / 100
[1] 3.59
```

The value returned by an R function is the argument of the `return` statement or the value of the last evaluated expression. The `sumdice` function could be written as

```
sumdice <- function(n)
    sum(sample(1:6, size=n, replace=TRUE))
```

Functions can have default argument values. For example, `sumdice` can be generalized to roll s-sided dice, but keep the default as 6-sided. The usage is shown below.

```
sumdice <- function(n, sides = 6) {
    if (sides < 1) return (0)
    k <- sample(1:sides, size=n, replace=TRUE)
    return(sum(k))
}

> sumdice(5)      #default 6 sides
[1] 12
> sumdice(n=5, sides=4)  #4 sides
[1] 14
```

The body of a function can be as short as one line, like the first `sumdice` function above, or have many lines. The function body must be enclosed in braces when it has more than one line.

An easy way to display the list of arguments to a function is `args`. Try for example `args(sample)`. If you have coded the function `sumdice` above, try `args(sumdice)`. ◇

1.8 Arrays, Data Frames, and Lists

Arrays, data frames, and lists are some of the objects used to store data in R. A matrix is a two-dimensional array. A data frame is not a matrix, although it can be represented in a rectangular layout like a matrix. Unlike a matrix, the columns of a data frame may be different types of variables. Arrays contain a single type.

Data Frames

A data frame is a list of variables, each of the same length but not necessarily of the same type. In this section we will discuss how to extract values of variables from a data frame.

Example 1.2 (Iris data)**.** The Fisher `iris` data set gives four measurements

on observations from three species of iris. The first few cases in the `iris` data are shown below.

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
```

The `iris` data is an example of a data frame object. It has 150 cases in rows and 5 variables in columns. After loading the data, variables can be referenced by $name (the column name), by subscripts like a matrix, or by position using the `[[ ]]` operator. The list of variable names is returned by `names`. Some examples with output are shown below.

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
> w <- iris[[2]]   #Sepal.Width
> mean(w)
[1] 3.057333
```

Alternately, the data frame can be attached and variables referenced directly by name. If a data frame is attached, it is a good practice to `detach` it when it is no longer needed, to avoid clashes with names of other variables.

```
> attach(iris)
> summary(Petal.Length[51:100]) #versicolor petal length
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.00    4.00    4.35    4.26    4.60    5.10
```

`with` and `by`

If we only need the `iris` data temporarily, we can use `with`. The syntax in this example would be

```
with(iris, summary(Petal.Length[51:100]))
```

However, `with` does not make changes outside of its local scope. It is best used for displaying or printing results. We can, however, assign the value of the evaluated expression to an object to save it.

```
out <- with(iris, summary(Petal.Length[51:100]))
```

Suppose we wish to compute the means of all variables, by species. The first four columns of the data frame can be extracted with `iris[,1:4]`. Here the missing row index indicates that all rows should be included. The `by` function easily computes the means by species.

```
> by(iris[,1:4], Species, colMeans)
Species: setosa
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
5.006        3.428        1.462        0.246
------------------------------------------------
Species: versicolor
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
5.936        2.770        4.260        1.326
------------------------------------------------
Species: virginica
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
6.588        2.974        5.552        2.026

> detach(iris)
```

◇

R Note 1.4

Although `iris$Sepal.Width`, `iris[[2]]`, and `iris[ ,2]` all produce the same result, the `$` and `[[ ]]` operators can only select one element, while the `[ ]` operator can select several. See the help topic `Extract`.

Arrays and Matrices

An array is a multiply subscripted collection of a single type of data. An array has a dimension attribute, which is a vector containing the dimensions of the array.

Example 1.3 (Arrays)**.** Different arrays are shown. The sequence of numbers from 1 to 24 is first a vector without a dimension attribute, then a one-dimensional array, then used to fill a 4 by 6 matrix, and finally a 3 by 4 by 2 array.

```
x <- 1:24                       # vector
dim(x) <- length(x)             # 1 dimensional array
matrix(1:24, nrow=4, ncol=6)    # 4 by 6 matrix
x <- array(1:24, c(3, 4, 2))    # 3 by 4 by 2 array
```

The $3 \times 4 \times 2$ array defined by the last statement is displayed below.

```
, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
, , 2
     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

The array `x` is displayed showing `x[, , 1]` (the first 3×4 elements) followed by `x[, , 2]` (the second 3×4 elements). ◇

A matrix is a doubly subscripted array of a single type of data. If `A` is a matrix, then `A[i, j]` is the ij-th element of `A`, `A[, j]` is the j-th column of `A`, and `A[i, ]` is the i-th row of `A`. A range of rows or columns can be extracted using the `:` sequence operator. For example, `A[2:3, 1:4]` extracts the 2×4 matrix containing rows 2 and 3 and columns 1 through 4 of `A`.

Example 1.4 (Matrices)**.** The statements

```
A <- matrix(0, nrow=2, ncol=2)
A <- matrix(c(0, 0, 0, 0), nrow=2, ncol=2)
A <- matrix(0, 2, 2)
```

all assign to `A` the 2×2 zero matrix. Matrices are filled in column major order by default; that is, the row index changes faster than the column index. Thus,

```
A <- matrix(1:8, nrow=2, ncol=4)
```

stores in `A` the matrix

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}.$$

If necessary, use the option `byrow=TRUE` in `matrix` to change the default. ◇

Example 1.5 (Iris data: Example 1.2, cont.)**.** We can convert the first four columns of the iris data to a matrix using `as.matrix`.

```
> x <- as.matrix(iris[,1:4]) #all rows of columns 1 to 4

> mean(x[,2])          #mean of sepal width, all species
[1] 3.057333
> mean(x[51:100,3])    #mean of petal length, versicolor
[1] 4.26
```

It is possible to convert the matrix to a three-dimensional array, but arrays (and matrices) are stored in "column major order" by default. For arrays, "column major" means that the indices to the left are changing faster than indices to the right. In this case it is easy to convert the matrix to a $50 \times 3 \times 4$ array, with the species as the second dimension. This works because in the data matrix, by column major order, the iris species changes faster than the variable name (column).

```
> y <- array(x, dim=c(50, 3, 4))
> mean(y[,,2])  #mean of sepal width, all species
[1] 3.057333
> mean(y[,2,3]) #mean of petal length, versicolor
[1] 4.26
```

It is somewhat more difficult to produce a $50 \times 4 \times 3$ array of iris data, with species as the third dimension. Here is one approach. First the matrix is sliced into three blocks of 50 observations each, corresponding to the three species. Then the three blocks are concatenated into a vector length 600, so that species is changing the slowest, and observation (row) is changing fastest. This vector then fills a $50 \times 4 \times 3$ array.

```
> y <- array(c(x[1:50,], x[51:100,], x[101:150,]), dim=c(50,4,3))
> mean(y[,2,])  #mean of sepal width, all species
[1] 3.057333
> mean(y[,3,2]) #mean of petal length, versicolor
[1] 4.26
```

This array is provided in R as the data set `iris3`. ◇

Lists

A list is an ordered collection of objects. The members of a list (the components) can be different types. Lists are more general than data frames; in fact, a data frame is a list with class "data.frame". A list can be created by the `list()` function.

Some functions return list objects. Two examples are shown below; the run length encoding function `rle` in Example 1.6 and the Wilcoxon test in Example 1.7.

Example 1.6 (Run length encoding)**.** Consider a coin flipping experiment. A "run" is a sequence of heads or tails. It is known that the maximum run length in a sequence of n Bernoulli trials ($p = 0.5$) should be about $log_2(n)$. The R function `rle` computes run lengths for a sequence of Bernoulli trials. We can simulate 1000 independent flips of a fair coin using the R Binomial random generator function `rbinom`.

```
  n <- 1000
  x <- rbinom(n, size = 1, prob = .5)
  table(x)
> x
  0   1
520 480
```

Here we can assign outcome 1 to heads and 0 to tails. We are interested in the pattern of runs of heads and tails; in particular, we are interested in the distribution of run lengths. The first part of the sequence can be shown with the `head` function.

```
> head(x, 30)
[1] 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 0 0 0 0 1 0
```

The value returned by `rle` has two components: lengths and values.

```
r <- rle(x)
> str(r)
List of 2
 $ lengths: int [1:483] 2 2 1 4 3 1 1 1 1 2 ...
 $ values : int [1:483] 0 1 0 1 0 1 0 1 0 1 ...
 - attr(*, "class")= chr "rle"
```

The R structure function `str` is very helpful when we want information about a list object. To extract one of the components, we can use the dollar sign and name of the component or double bracket and position.

```
> head(r$lengths)
[1] 2 2 1 4 3 1
> head(r[[1]])
[1] 2 2 1 4 3 1
```

Is the maximum run length in this example approximately equal to $\log_2(n)$?

```
> max(r$lengths)
[1] 10
> log2(length(x))
[1] 9.965784
```

◇

Lists are frequently used to return several results of a function in a single object. Several classical hypothesis tests that return class `htest` are a good example. See for example the help topic for `t.test` or `chisq.test`. Refer to the "Value" section of the documentation. The value returned is a list containing the test statistic, p-value, etc. The components of a list can be referenced by name using $ or by position using `[[ ]]`.

Example 1.7 (Named list)**.** The Wilcoxon rank sum test is implemented in the function `wilcox.test`. Here the test is applied to two normal samples with different means.

```
w <- wilcox.test(rnorm(10), rnorm(10, 2))
> w    #print the summary

        Wilcoxon rank sum test

data:  rnorm(10) and rnorm(10, 2)
W = 2, p-value = 4.33e-05
alternative hypothesis:
true location shift is not equal to 0
```

```
> w$statistic       #stored in object w
W 2
> w$p.value
[1] 4.330035e-05
```

Try `unlist(w)` and `unclass(w)` to see more details. ◇

Some examples of functions in this book that return a named list can be found in Examples 8.13, 12.12, and 14.7.

Example 1.8 (A `list` of names)**.** Below we create a `list` to assign row and column names in a matrix. The first component for row names will be `NULL` in this case because we do not want to assign row names.

```
a <- matrix(runif(8), 4, 2)   #a 4x2 matrix
dimnames(a) <- list(NULL, c("x", "y"))
```

Here is the 4×2 matrix with column names (type `a` to display it).

```
             x         y
[1,] 0.88009604 0.6583918
[2,] 0.32964955 0.1385332
[3,] 0.61625490 0.1378254
[4,] 0.08102034 0.1746324

# if we want row names
> dimnames(a) <- list(letters[1:4], c("x", "y"))
> a
           x         y
a 0.88009604 0.6583918
b 0.32964955 0.1385332
c 0.61625490 0.1378254
d 0.08102034 0.1746324

# another way to assign row names
> row.names(a) <- list("NE", "NW", "SW", "SE")
> a
            x         y
NE 0.88009604 0.6583918
NW 0.32964955 0.1385332
SW 0.61625490 0.1378254
SE 0.08102034 0.1746324
```

◇

1.9 Formula Specification

Some functions in R take a `formula` object as an argument. Examples include the function to fit linear models (`lm`) and certain graphics functions like `boxplot`. For example, a formula for simple linear regression of response variable `y` on a single predictor `x` is specified by `y ~ x`, which represents the model $y = \beta_0 + \beta_1 x + \varepsilon$. To specify the regression model $y = \beta_1 x + \varepsilon$, without an intercept term, the formula is `y ~ 0 + x`. For example, compare the following regression models for the `rock` data:

```
lm(rock$peri ~ rock$area)
lm(rock$peri ~ 0 + rock$area)
lm(rock$peri ~ 1 + rock$area)
```

The `formula` syntax can represent more complicated models with several terms, interactions, etc. The syntax is based on Wilkinson's notation [319]. See Hastie [44, Section 2.2] for its implementation in S and R languages, or [198] for an online version of documentation for Wilkinson notation.

In Section 1.10, parallel boxplots are generated using the formula argument to `boxplot`. Several examples of formulas for linear models are in Section 8.5 and Chapter 9.

1.10 Graphics

The R `graphics` package contains most of the commonly used graphics functions. In this section, for reference, some of the graphics functions and options or parameters are listed. Examples of graphics and the R code used to produce them appear throughout the text. See Chang [47] and Murrell [213] for many more examples. Maindonald and Braun [189]), and Venables and Ripley [293] also have many examples of graphics in R.

Table 1.4 lists some basic 2D graphics functions in R (`graphics`) and other packages. Several examples using the graphics functions in Table 1.4 are given throughout the text. See Table 5.1 and the examples of Chapter 5 for more 2D graphics functions and some 3D visualization methods.