Modern Data Science with R
CHAPMAN & HALL/CRC
Texts in Statistical Science Series
Series Editors
Francesca Dominici, Harvard School of Public Health, USA
Julian J. Faraway, University of Bath, UK
Martin Tanner, Northwestern University, USA
Jim Zidek, University of British Columbia, Canada

Statistical Theory: A Concise Introduction
F. Abramovich and Y. Ritov
Practical Multivariate Analysis, Fifth Edition
A. Affi, S. May, and V.A. Clark
Practical Statistics for Medical Research
D.G. Altman
Interpreting Data: A First Course in Statistics
A.J.B. Anderson
Introduction to Probability with R
K. Baclawski
Linear Algebra and Matrix Analysis for Statistics
S. Banerjee and A. Roy
Modern Data Science with R
B. S. Baumber, D. T. Kaplan, and N. J. Horton
P. J. Bickel and K. A. Doksum
Mathematical Statistics: Basic Ideas and Selected Topics, Volume II
P. J. Bickel and K. A. Doksum
Analysis of Categorical Data with R
C. R. Bilder and T. M. Loughin
Statistical Methods for SPC and TQM
D. Bissell
Introduction to Probability
J. K. Blitzstein and J. Hwang
Bayesian Methods for Data Analysis, Third Edition
B. P. Carlin and T. A. Louis
Second Edition
R. Caulcutt
The Analysis of Time Series: An Introduction, Sixth Edition
C. Chatfield
Introduction to Multivariate Analysis
C. Chatfield and A. J. Collins
C. Chatfield
C. Chatfield
Analysis of Variance, Design, and Regression: Linear Modeling for Unbalanced Data, Second Edition
R. Christensen
Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians
R. Christensen, W. Johnson, A. Branscum, and T.E. Hanson
Modelling Binary Data, Second Edition
D. Collett
Modelling Survival Data in Medical Research, Third Edition
D. Collett
Introduction to Statistical Methods for Clinical Trials
T.D. Cook and D.L. DeMets
Applied Statistics: Principles and Examples
D.R. Cox and E.J. Snell
Multivariate Survival Analysis and Competing Risks
M. Crowder
Statistical Analysis of Reliability Data
M.J. Crowder, A.C. Kimber, T.J. Sweeting, and R.L. Smith
An Introduction to Generalized Linear Models, Third Edition
A.J. Dobson and A.G. Barnett
Nonlinear Time Series: Theory, Methods, and Applications with R Examples
R. Douc, E. Moulines, and D.S. Stoffer
Introduction to Optimization Methods and Their Applications in Statistics
B.S. Everitt
Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models, Second Edition
J.J. Faraway
Linear Models with R, Second Edition
J.J. Faraway
H. S. Migon, D. Gamerman, and F. Louzada

Beyond ANOVA: Basics of Applied Statistics
R.G. Miller, Jr.

A Primer on Linear Models
J.F. Monahan

Stochastic Processes: From Applications to Theory
P.D. Moral and S. Penev

Applied Stochastic Modelling, Second Edition
B.J.T. Morgan

Elements of Simulation
B.J.T. Morgan

Probability: Methods and Measurement
A. O'Hagan

Introduction to Statistical Limit Theory
A.M. Polansky

Applied Bayesian Forecasting and Time Series Analysis
A. Pole, M. West, and J. Harrison

Statistics in Research and Development, Time Series: Modeling, Computation, and Inference
R. Prado and M. West

Essentials of Probability Theory for Statisticians
M.A. Proschan and P.A. Shaw

Introduction to Statistical Process Control
P. Qiu

Sampling Methodologies with Applications
P.S.R.S. Rao

A First Course in Linear Model Theory
N. Ravishankar and D.K. Dey

Essential Statistics, Fourth Edition
D.A.G. Rees

Stochastic Modeling and Mathematical Statistics: A Text for Statisticians and Quantitative Scientists
F.J. Samaniego

Statistical Methods for Spatial Data Analysis
O. Schabenberger and C.A. Gotway

Bayesian Networks: With Examples in R
M. Scutari and J.-B. Denis

Large Sample Methods in Statistics
P.K. Sen and J. da Motta Singer

Spatio-Temporal Methods in Environmental Epidemiology
G. Shaddick and J.V. Zidek

Decision Analysis: A Bayesian Approach
J.Q. Smith

Analysis of Failure and Survival Data
P.J. Smith

Applied Statistics: Handbook of GENSTAT Analyses
E.J. Snell and H. Simpson

Applied Nonparametric Statistical Methods, Fourth Edition
P. Sprent and N.C. Smeeton

Data Driven Statistical Methods
P. Sprent

Generalized Linear Mixed Models: Modern Concepts, Methods and Applications
W.W. Stroup

Survival Analysis Using S: Analysis of Time-to-Event Data
M. Tableman and J.S. Kim

Applied Categorical and Count Data Analysis
W. Tang, H. He, and X.M. Tu

Elementary Applications of Probability Theory, Second Edition
H.C. Tuckwell

Introduction to Statistical Inference and Its Applications with R
M.W. Trosset

Understanding Advanced Statistical Methods
P.H. Westfall and K.S.S. Henning

G.B. Wetherill and D.W. Brown

Generalized Additive Models: An Introduction with R
S. Wood

Epidemiology: Study Design and Data Analysis, Third Edition
M. Woodward

Practical Data Analysis for Designed Experiments
B.S. Yandell
## Contents

List of Tables xv
List of Figures xvii
Preface xxiii

I Introduction to Data Science 1

1 Prologue: Why data science? 3
  1.1 What is data science? 4
  1.2 Case study: The evolution of sabermetrics 6
  1.3 Datasets 7
  1.4 Further resources 8

2 Data visualization 9
  2.1 The 2012 federal election cycle 9
    2.1.1 Are these two groups different? 10
    2.1.2 Graphing variation 11
    2.1.3 Examining relationships among variables 12
    2.1.4 Networks 13
  2.2 Composing data graphics 14
    2.2.1 A taxonomy for data graphics 14
    2.2.2 Color 19
    2.2.3 Dissecting data graphics 20
  2.3 Importance of data graphics: Challenger 23
  2.4 Creating effective presentations 27
  2.5 The wider world of data visualization 28
  2.6 Further resources 30
  2.7 Exercises 30

3 A grammar for graphics 33
  3.1 A grammar for data graphics 33
    3.1.1 Aesthetics 34
    3.1.2 Scale 37
    3.1.3 Guides 38
    3.1.4 Facets 38
    3.1.5 Layers 38
  3.2 Canonical data graphics in R 39
    3.2.1 Univariate displays 39
## CONTENTS

3.2.2 Multivariate displays ........................................ 43  
3.2.3 Maps .................................................. 48  
3.2.4 Networks .............................................. 48  
3.3 Extended example: Historical baby names .................. 48  
3.3.1 Percentage of people alive today ....................... 50  
3.3.2 Most common women's names ............................ 56  
3.4 Further resources .......................................... 58  
3.5 Exercises ................................................ 58  

4 Data wrangling ................................................... 63  
4.1 A grammar for data wrangling ............................... 63  
4.1.1 `select()` and `filter()` ............................... 63  
4.1.2 `mutate()` and `rename()` ........................... 66  
4.1.3 `arrange()` ............................................. 69  
4.1.4 `summarize()` with `group_by()` ....................... 70  
4.2 Extended example: Ben's time with the Mets ............. 72  
4.3 Combining multiple tables .................................. 79  
4.3.1 `inner_join()` ......................................... 79  
4.3.2 `left_join()` .......................................... 81  
4.4 Extended example: Manny Ramirez ......................... 82  
4.5 Further resources .......................................... 88  
4.6 Exercises ................................................ 88  

5 Tidy data and iteration ......................................... 91  
5.1 Tidy data .................................................. 91  
5.1.1 Motivation ............................................. 91  
5.1.2 What are tidy data? ..................................... 93  
5.2 Reshaping data .............................................. 98  
5.2.1 Data verbs for converting wide to narrow and vice versa ...................... 100  
5.2.2 Spreading ............................................... 100  
5.2.3 Gathering .............................................. 101  
5.2.4 Example: Gender-neutral names ....................... 101  
5.3 Naming conventions ........................................ 103  
5.4 Automation and iteration .................................. 104  
5.4.1 Vectorized operations ................................. 104  
5.4.2 The `apply()` family of functions .................. 106  
5.4.3 Iteration over subgroups with `dplyr::do()` ......... 110  
5.4.4 Iteration with `mosaic::do` ........................... 113  
5.5 Data intake ............................................... 116  
5.5.1 Data-table friendly formats .......................... 116  
5.5.2 APIs .................................................. 120  
5.5.3 Cleaning data ......................................... 120  
5.5.4 Example: Japanese nuclear reactors ................. 126  
5.6 Further resources .......................................... 127  
5.7 Exercises ................................................ 128  

6 Professional Ethics ............................................ 131  
6.1 Introduction ............................................... 131  
6.2 Truthful falsehoods ....................................... 131  
6.3 Some settings for professional ethics .................... 134  
6.3.1 The chief executive officer ........................... 134
CONTENTS

6.3.2 Employment discrimination ............................ 134
6.3.3 Data scraping ........................................ 135
6.3.4 Reproducible spreadsheet analysis ................... 135
6.3.5 Drug dangers .......................................... 135
6.3.6 Legal negotiations ...................................... 136
6.4 Some principles to guide ethical action .................. 136
6.4.1 Applying the precepts ................................ 137
6.5 Data and disclosure ....................................... 140
6.5.1 Reidentification and disclosure avoidance .......... 140
6.5.2 Safe data storage .................................... 141
6.5.3 Data scraping and terms of use ...................... 141
6.6 Reproducibility ........................................... 142
6.6.1 Example: Erroneous data merging ................... 142
6.7 Professional guidelines for ethical conduct ............ 143
6.8 Ethics, collectively ....................................... 143
6.9 Further resources ........................................ 144
6.10 Exercises .................................................. 144

II Statistics and Modeling 147

7 Statistical foundations 149
7.1 Samples and populations .................................. 149
7.2 Sample statistics ......................................... 152
7.3 The bootstrap ............................................. 155
7.4 Outliers .................................................... 157
7.5 Statistical models: Explaining variation ............... 159
7.6 Confounding and accounting for other factors .......... 162
7.7 The perils of p-values ................................... 165
7.8 Further resources ........................................ 167
7.9 Exercises .................................................. 168

8 Statistical learning and predictive analytics 171
8.1 Supervised learning ....................................... 172
8.2 Classifiers ................................................. 173
8.2.1 Decision trees ........................................ 173
8.2.2 Example: High-earners in the 1994 United States Census ... 174
8.2.3 Tuning parameters .................................... 180
8.2.4 Random forests ....................................... 181
8.2.5 Nearest neighbor ...................................... 182
8.2.6 Naïve Bayes ........................................... 183
8.2.7 Artificial neural networks ............................. 185
8.3 Ensemble methods ........................................ 186
8.4 Evaluating models ........................................ 188
8.4.1 Cross-validation ...................................... 188
8.4.2 Measuring prediction error ......................... 189
8.4.3 Confusion matrix ..................................... 189
8.4.4 ROC curves ........................................... 189
8.4.5 Bias-variance trade-off ............................... 192
8.4.6 Example: Evaluation of income models .......... 192
8.5 Extended example: Who has diabetes? .................. 196
CONTENTS

8.6 Regularization ........................................ 201
8.7 Further resources .................................... 201
8.8 Exercises ............................................. 201

9 Unsupervised learning ................................. 205
9.1 Clustering ............................................ 205
  9.1.1 Hierarchical clustering ......................... 206
  9.1.2 k-means ......................................... 210
9.2 Dimension reduction ................................. 211
  9.2.1 Intuitive approaches ......................... 212
  9.2.2 Singular value decomposition ............... 213
9.3 Further resources ................................ 218
9.4 Exercises ............................................. 218

10 Simulation ............................................ 221
10.1 Reasoning in reverse ............................. 221
10.2 Extended example: Grouping cancers .......... 222
10.3 Randomizing functions ......................... 223
10.4 Simulating variability ............................ 225
  10.4.1 The partially planned rendezvous .......... 225
  10.4.2 The jobs report .............................. 227
  10.4.3 Restaurant health and sanitation grades .. 228
10.5 Simulating a complex system ................. 231
10.6 Random networks ................................ 233
10.7 Key principles of simulation .................. 233
10.8 Further resources ................................ 235
10.9 Exercises ............................................. 236

III Topics in Data Science .............................. 241

11 Interactive data graphics ......................... 243
  11.1 Rich Web content using D3.js and htmlwidgets ........................................ 243
  11.1.1 Leaflet ......................................... 244
  11.1.2 Plot.ly ......................................... 244
  11.1.3 DataTables ..................................... 244
  11.1.4 dygraphs ...................................... 246
  11.1.5 streamgraphs ................................. 246
  11.2 Dynamic visualization using ggvis .......... 246
11.3 Interactive Web apps with Shiny ............... 247
11.4 Further customization ............................ 250
11.5 Extended example: Hot dog eating ............ 254
11.6 Further resources ................................ 258
11.7 Exercises ............................................. 258

12 Database querying using SQL ................. 261
  12.1 From dplyr to SQL ............................... 261
  12.2 Flat-file databases ............................... 265
  12.3 The SQL universe ................................ 266
  12.4 The SQL data manipulation language ....... 267
  12.4.1 SELECT...FROM .............................. 270
## CONTENTS

12.4.2 WHERE ........................................... 272
12.4.3 GROUP BY ................................. 275
12.4.4 ORDER BY ................................. 277
12.4.5 HAVING ..................................... 278
12.4.6 LIMIT ....................................... 280
12.4.7 JOIN .......................................... 281
12.4.8 UNION ......................................... 286
12.4.9 Subqueries .................................. 287
12.5 Extended example: FiveThirtyEight flights 289
12.6 SQL vs. R ........................................ 298
12.7 Further resources ................................. 298
12.8 Exercises ........................................ 298

13 Database administration 301
13.1 Constructing efficient SQL databases ............ 301
13.1.1 Creating new databases ....................... 301
13.1.2 CREATE TABLE .................................. 302
13.1.3 Keys ........................................... 303
13.1.4 Indices ........................................ 304
13.1.5 EXPLAIN ...................................... 306
13.1.6 Partitioning ................................... 308
13.2 Changing SQL data ................................ 308
13.2.1 UPDATE ...................................... 308
13.2.2 INSERT ....................................... 309
13.2.3 LOAD DATA .................................. 309
13.3 Extended example: Building a database .......... 309
13.3.1 Extract ....................................... 310
13.3.2 Transform .................................... 310
13.3.3 Load into MySQL database .................... 310
13.4 Scalability ....................................... 314
13.5 Further resources ................................ 314
13.6 Exercises ......................................... 314

14 Working with spatial data 317
14.1 Motivation: What’s so great about spatial data? .... 317
14.2 Spatial data structures ............................. 319
14.3 Making maps .................................... 322
14.3.1 Static maps with ggmap ...................... 322
14.3.2 Projections .................................. 324
14.3.3 Geocoding, routes, and distances .......... 330
14.3.4 Dynamic maps with leaflet .................... 332
14.4 Extended example: Congressional districts ....... 333
14.4.1 Election results ................................ 334
14.4.2 Congressional districts ....................... 336
14.4.3 Putting it all together ......................... 338
14.4.4 Using ggmap .................................. 340
14.4.5 Using leaflet .................................. 343
14.5 Effective maps: How (not) to lie .................. 343
14.6 Extended example: Historical airline route maps ...... 345
14.6.1 Using ggmap .................................. 346
14.6.2 Using leaflet .................................. 347
CONTENTS

14.7 Projecting polygons ........................................... 349
14.8 Playing well with others ..................................... 351
14.9 Further resources ............................................. 352
14.10 Exercises ...................................................... 352

15 Text as data ....................................................... 355
15.1 Tools for working with text .................................. 355
  15.1.1 Regular expressions using Macbeth ......................... 355
  15.1.2 Example: Life and death in Macbeth ......................... 359
15.2 Analyzing textual data ....................................... 360
  15.2.1 Corpora .................................................... 364
  15.2.2 Word clouds .............................................. 365
  15.2.3 Document term matrices ................................ 365
15.3 Ingesting text ................................................ 367
  15.3.1 Example: Scraping the songs of the Beatles ............... 367
  15.3.2 Scraping data from Twitter ................................ 369
15.4 Further resources ............................................ 374
15.5 Exercises ...................................................... 374

16 Network science ................................................ 377
16.1 Introduction to network science ............................... 377
  16.1.1 Definitions ................................................. 377
  16.1.2 A brief history of network science ......................... 378
16.2 Extended example: Six degrees of Kristen Stewart ........... 382
  16.2.1 Collecting Hollywood data ................................. 382
  16.2.2 Building the Hollywood network ......................... 384
  16.2.3 Building a Kristen Stewart oracle ......................... 387
16.3 PageRank ..................................................... 390
16.4 Extended example: 1996 men's college basketball .......... 391
16.5 Further resources ............................................ 398
16.6 Exercises ...................................................... 398

17 Epilogue: Towards “big data” .................................. 401
17.1 Notions of big data ........................................... 401
17.2 Tools for bigger data ......................................... 403
  17.2.1 Data and memory structures for big data .................. 403
  17.2.2 Compilation ............................................... 404
  17.2.3 Parallel and distributed computing ....................... 404
  17.2.4 Alternatives to SQL ...................................... 411
17.3 Alternatives to R .............................................. 413
17.4 Closing thoughts ............................................. 413
17.5 Further thoughts .............................................. 413

IV Appendices ...................................................... 415
A Packages used in this book ................................... 417
  A.1 The mdsr package ........................................... 417
  A.2 The etl package suite ....................................... 417
  A.3 Other packages .............................................. 418
  A.4 Further resources ........................................... 420
CONTENTS

E.2 Multiple regression ................................................. 470
  E.2.1 Parallel slopes: Multiple regression with a categorical
       variable ....................................................... 470
  E.2.2 Parallel planes: Multiple regression with a second
       quantitative variable ........................................ 471
  E.2.3 Non-parallel slopes: Multiple regression with interaction .... 472
  E.2.4 Modelling non-linear relationships .......................... 472
E.3 Inference for regression ........................................... 474
E.4 Assumptions underlying regression .............................. 475
E.5 Logistic regression ................................................. 477
E.6 Further resources .................................................. 481
E.7 Exercises ........................................................... 482

F Setting up a database server 487
  F.1 SQLite .......................................................... 487
  F.2 MySQL .......................................................... 488
    F.2.1 Installation ............................................... 488
    F.2.2 Access .................................................... 488
    F.2.3 Running scripts from the command line ................. 491
  F.3 PostgreSQL .................................................... 491
  F.4 Connecting to SQL .............................................. 492
    F.4.1 The command line client .................................. 492
    F.4.2 GUIs ...................................................... 492
    F.4.3 R and RStudio ............................................. 492
    F.4.4 Load into SQLite database ............................... 497

Bibliography 499

Indices 513
  Subject index ....................................................... 514
  R index ........................................................... 543
List of Tables

3.1 A selection of variables from the first six rows of the CIACountries data table. 34
3.2 Glyph-ready data for the barplot layer in Figure 3.7. 39
3.3 Table of canonical data graphics and their corresponding ggplot2 commands. Note that mosaicplot() is not part of the ggplot2 package. 47

5.1 A data table showing how many babies were given each name in each year in the U.S., for a few names. 93
5.2 The most popular baby names across all years. 94
5.3 Ward and precinct votes cast in the 2013 Minneapolis mayoral election. 95
5.4 A selection from the Minneapolis election data in tidy form. 96
5.5 Individual ballots in the Minneapolis election. Each voter votes in one ward in one precinct. The ballot marks the voter’s first three choices for mayor. 97
5.6 An excerpt of runners’ performance over time in a 10-mile race. 98
5.7 BP-wide: a data table in a wide format. 99
5.8 BP-narrow: a tidy data table in a narrow format. 100
5.9 A data table extending the information in Tables 5.8 and 5.7 to include additional variables and repeated measurements. The narrow format facilitates including new cases or variables. 100
5.10 The third table embedded in the Wikipedia page on running records. 119
5.11 The fourth table embedded in the Wikipedia page on running records. 120
5.12 Four of the variables from the houses-for-sale.csv file giving features of the Saratoga houses stored as integer codes. Each case is a different house. 121
5.13 The Translations data table rendered in a wide format. 121
5.14 The Houses data with re-coded categorical variables. 122
5.15 Starting and ending dates for each transcriber involved in the OrdwayBirds project. 124

9.1 Sample voting records data from the Scottish Parliament. 212

12.1 Equivalent commands in SQL and R, where a and b are SQL tables and R data.frames. 270

14.1 Hypothetical data from 1854 cholera outbreak. 318

A.1 List of packages used in this book. Most packages are available on CRAN. Packages available from GitHub include: airlines, fec, imdb, sparklyr, and streamgraph. 420

B.1 Some of the interactive courses available within swirl. 426
B.2 A complete list of CRAN task views. 437
List of Figures

1.1 Excerpt from Graunt’s bills of mortality. .......................... 4
2.1 Amount of money spent on individual candidates in the general election phase of the 2012 federal election cycle, in millions of dollars ............... 10
2.2 Amount of money spent on individual candidates in the general election phase of the 2012 federal election cycle, in millions of dollars, broken down by type of spending .................................................. 11
2.3 Amount of money spent on individual candidacies by political party affiliation during the general election phase of the 2012 federal election cycle ............................................................... 12
2.4 Amount of money spent on individual candidacies by political party affiliation during the general election phase of the 2012 federal election cycle, broken down by office being sought .................................................. 13
2.5 Donations made by individuals to the PACs supporting the two major presidential candidates in the 2012 election .................................................. 14
2.6 Donations made by individuals to the PACs supporting the two major presidential candidates in the 2012 election, separated by election phase .... 15
2.7 Scatterplot illustrating the relationship between number of dollars spent supporting and number of votes earned by Democrats in 2012 elections for the House of Representatives .................................................. 16
2.8 Scatterplot illustrating the relationship between percentage of dollars spent supporting and percentage of votes earned by Democrats in the 2012 House of Representatives elections .................................................. 16
2.9 Campaign funding network for candidates from Massachusetts, 2012 federal elections ........................................................................................................................................................................................................................................................................................................ 17
2.10 Diverging red-blue color palette ........................................ 20
2.11 Palettes available through the RColorBrewer package ........ 21
2.12 Bar graph of average SAT scores among states with at least two-thirds of students taking the test .................................................................................. 22
2.13 Scatterplot of world record time in 100-meter freestyle swimming. 23
2.14 Pie charts showing the breakdown of substance of abuse among HELP study participants, faceted by homeless status .................................................. 24
2.15 Choropleth map of population among Massachusetts Census tracts, based on 2010 U.S. Census ...................................................................................................................... 25
2.16 A scatterplot with smoother demonstrating the relationship between temperature and O-ring damage on solid rocket motors. The dots are semitransparent, so that darker dots indicate multiple observations with the same values. ...................................................................................................................... 25
2.17 A recreation of Tufte’s scatterplot demonstrating the relationship between temperature and O-ring damage on solid rocket motors . . . . . . . 26
# LIST OF FIGURES

2.18 Reprints of two Morton Thiokol data graphics. [195] .......................... 27
2.19 Still images from *Forms*, by Memo Akten and Quayola. Each image represents an athletic movement made by a competitor at the Commonwealth Games, but reimagined as a collection of moving 3D digital objects. Reprinted with permission. .................................................. 29

3.1 Scatterplot using only the position aesthetic for glyphs. ....................... 35
3.2 Scatterplot in which net_users is mapped to color. ............................ 35
3.3 Scatterplot using both location and label as aesthetics. ...................... 36
3.4 Scatterplot in which net_users is mapped to color and educ mapped to size. Compare this graphic to Figure 3.6, which displays the same data using facets. 36
3.5 Scatterplot using a logarithmic transformation of GDP that helps to mitigate visual clustering caused by the right-skewed distribution of GDP among countries. .................................................. 37
3.6 Scatterplot using facets for different ranges of Internet connectivity. ..... 38
3.7 Bar graph of average charges for medical procedures in New Jersey. ..... 40
3.8 Bar graph adding a second layer to provide a comparison of New Jersey to other states. Each dot represents one state, while the bars represent New Jersey. .................................................. 40
3.9 Histogram showing the distribution of Math SAT scores by state. .......... 41
3.10 Density plot showing the distribution of Math SAT scores by state. ....... 42
3.11 A bar plot showing the distribution of Math SAT scores for a selection of states. .................................................................................. 42
3.12 A stacked bar plot showing the distribution of substance of abuse for participants in the HELP study. Compare this to Figure 2.14. .............. 43
3.13 Scatterplot using the *color* aesthetic to separate the relationship between two numeric variables by a third categorical variable. .............. 44
3.14 Scatterplot using a *facet_wrap()* to separate the relationship between two numeric variables by a third categorical variable. ................. 45
3.15 A scatterplot for 1,000 random individuals from the NHANES study. Note how mapping gender to color illuminates the differences in height between men and women. .................................. 45
3.16 A time series showing the change in temperature at the MacLeish field station in 2015. ........................................................................ 46
3.17 A box-and-whisker plot showing the distribution of foot length by gender for 39 children. ................................................................. 47
3.18 Mosaic plot (ekkosogram) of diabetes by age and weight status (BMI). 47
3.19 A choropleth map displaying oil production by countries around the world in barrels per day .......................................................... 48
3.20 A network diagram displaying the relationship between types of cancer cell lines ........................................................................... 49
3.21 Popularity of the name “Joseph” as constructed by FiveThirtyEight. .... 50
3.22 Recreation of the age distribution of “Joseph” plot .............................. 53
3.23 Age distribution of American girls named “Josephine” ....................... 54
3.24 Comparison of the name “Jessie” across two genders ....................... 54
3.25 Gender breakdown for the three most “unisex” names ...................... 55
3.26 Gender breakdown for the three most “unisex” names, oriented vertically .... 55
3.27 FiveThirtyEight’s depiction of the age ranges for the 25 most common female names. ................................................................. 57
3.28 Recreation of FiveThirtyEight’s plot of the age distributions for the 25 most common women’s names .................................................. 62
LIST OF FIGURES

4.1 The `filter()` function. At left, a data frame that contains matching entries in a certain column for only a subset of the rows. At right, the resulting data frame after filtering. ........................................ 64
4.2 The `select()` function. At left, a data frame, from which we retrieve only a few of the columns. At right, the resulting data frame after selecting those columns. .................................................. 64
4.3 The `mutate()` function. At left, a data frame. At right, the resulting data frame after adding a new column. .......................... 66
4.4 The `arrange()` function. At left, a data frame with an ordinal variable. At right, the resulting data frame after sorting the rows in descending order of that variable. ........................................ 69
4.5 The `summarize()` function. At left, a data frame. At right, the resulting data frame after aggregating three of the columns. ............ 70

5.1 A graphical depiction of voter turnout in the different wards ........... 96
5.2 Part of the codebook for the HELPrct data table from the mosaicData package. 99
5.3 Fit for the Pythagorean Winning Percentage model for all teams since 1954 111
5.4 Number of home runs hit by the team with the most home runs, 1916–2014 113
5.5 Distribution of best-fitting exponent across single seasons from 1961–2014 114
5.6 Bootstrap distribution of mean optimal exponent .......................... 115
5.7 Part of a page on mile-run world records from Wikipedia. Two separate data tables are visible. You can't tell from this small part of the page, but there are seven tables altogether on the page. These two are the third and fourth in the page. ........................................ 119
5.8 The transcribers of OrdwayBirds from lab notebooks worked during different time intervals ........................................ 124
5.9 Screenshot of Wikipedia’s list of Japanese nuclear reactors. ............. 126
5.10 Distribution of capacity of Japanese nuclear power plants over time .... 128

6.1 Reproduction of a data graphic reporting the number of gun deaths in Florida over time ................................. 132
6.2 A tweet by National Review on December 14, 2015 showing the change in global temperature over time. ................................. 133

7.1 The sampling distribution of the mean arrival delay with a sample size of $n = 25$ (left) and also for a larger sample size of $n = 100$ (right). ...... 154
7.2 Distribution of flight arrival delays in 2013 for flights to San Francisco from NYC airports that were delayed less than seven hours. The distribution features a long right tail (even after pruning the outliers). .......... 159
7.3 Association of flight arrival delays with scheduled departure time for flights to San Francisco from New York airports in 2013. .............. 160
7.4 Scatterplot of average SAT scores versus average teacher salaries (in thousands of dollars) for the 50 United States in 2010. .............. 163
7.5 Scatterplot of average SAT scores versus average teacher salaries (in thousands of dollars) for the 50 United States in 2010, stratified by the percentage of students taking the SAT in each state. ..................... 164
LIST OF FIGURES

8.1 A single partition of the census data set using the capital_gain variable to determine the split. Color, and the vertical line at $5,095.50 in capital gains tax indicate the split. If one paid more than this amount, one almost certainly made more than $50,000 in income. On the other hand, if one paid less than this amount in capital gains, one almost certainly made less than $50,000. ................................................................. 177
8.2 Decision tree for income using the census data ........................................... 178
8.3 Graphical depiction of the full recursive partitioning decision tree classifier 179
8.4 Performance of nearest neighbor classifier for different choices of k on census training data .......................................................... 184
8.5 Visualization of an artificial neural network .............................................. 187
8.6 ROC curve for naive Bayes model ............................................................ 191
8.7 Performance of nearest neighbor classifier for different choices of k on census training and testing data ................................................. 193
8.8 Comparison of ROC curves across five models on the Census testing data . 197
8.9 Illustration of decision tree for diabetes .................................................. 198
8.10 Scatterplot of age against BMI for individuals in the NHANES data set ... 199
8.11 Comparison of predictive models in the data space .................................. 202

9.1 An evolutionary tree for mammals. Source: [92] ........................................ 206
9.2 Distances between some US cities ........................................................... 208
9.3 A dendrogram constructed by hierarchical clustering from car-to-car distances implied by the Toyota fuel economy data ......................... 209
9.4 The world’s 4,000 largest cities, clustered by the 6-means clustering algorithm211
9.5 Visualization of the Scottish Parliament votes ......................................... 213
9.6 Scottish Parliament votes for two ballots .................................................... 214
9.7 Scatterplot showing the correlation between Scottish Parliament votes in two arbitrary collections of ballots .............................................. 215
9.8 Clustering members of Scottish Parliament based on SVD along the members216
9.9 Clustering of Scottish Parliament ballots based on SVD along the ballots . 217
9.10 Illustration of the Scottish Parliament votes when ordered by the primary vector of the SVD .............................................................. 218

10.1 Comparing the variation in expression for individual probes across cell lines in the NCI60 data (blue) and a simulation of a null hypothesis (red) ........ 224
10.2 Distribution of Sally and Joan arrival times (shaded area indicates where they meet) ................................................................. 229
10.3 True number of new jobs from simulation as well as three realizations from a simulation .............................................................. 229
10.4 Distribution of NYC restaurant health violation scores ............................ 230
10.5 Distribution of health violation scores under a randomization procedure ... 231
10.6 Convergence of the estimate of the proportion of times that Sally and Joan meet. ............................................................................. 235

11.1 ggplot2 depiction of the frequency of Beatles names over time ............... 245
11.2 A screenshot of the interactive plot of the frequency of Beatles names over time ..................................................................................... 245
11.3 A screenshot of the output of the DataTables package applied to the Beatles names. ............................................................................. 246
LIST OF FIGURES

11.4 A screenshot of the dygraphs display of the popularity of Beatles names over time. In this screenshot, the years range from 1940 to 1980, but in the live version, one can expand or contract that timespan ........................................... 247
11.5 A screenshot of the streamgraph display of Beatles names over time ................................................................. 248
11.6 A screenshot of the ggvis display of the proportion and number of male babies named “John” over time ................................................................. 249
11.7 A screenshot of the Shiny app displaying babies with Beatles names ................................................................. 250
11.8 Comparison of two ggplot2 themes .................................................................................................................. 252
11.9 Beatles plot with custom ggplot2 theme ........................................................................................................ 252
11.10 Beatles plot with customized mdsr theme ........................................................................................................ 253
11.11 Prevalence of Beatles names drawn in the style of an xkcd Web comic .......................................................... 254
11.12 Nathan Yau’s Hot Dog Eating data graphic (reprinted with permission from flowingdata.com) ................................................................. 255
11.13 A simple bar graph of hot dog eating ............................................................................................................... 256
11.14 Recreating the hot dog graphic in R ................................................................................................................ 258

12.1 FiveThirtyEight data graphic summarizing airline delays by carrier. Reproduced with permission ................................................................. 291
12.2 Re-creation of the FiveThirtyEight plot on flight delays ................................................................................ 294

14.1 John Snow’s original map of the 1854 Broad Street cholera outbreak. Source: Wikipedia ........................................................................................................... 319
14.2 A simple ggplot2 of the cholera deaths, with no context provided ........................................................................... 322
14.3 A modern-day map of the area surrounding Broad Street in London ................................................................................. 323
14.4 The world according to the Mercator (left) and Gall–Peters (right) projections ............................... 325
14.5 The contiguous United States according to the Lambert conformal conic (left) and Albers equal area (right) projections .................................................................................. 326
14.6 Erroneous reproduction of John Snow’s original map of the 1854 cholera outbreak ........................................................................................................ 328
14.7 Reproduction of John Snow’s original map of the 1854 cholera outbreak .............................................................. 329
14.8 The fastest route from Smith College to Amherst College ...................................................................................... 331
14.9 Alternative commuting routes from Ben’s old apartment in Brooklyn to Citi Field ........................................ 332
14.10 Static image from a leaflet plot of the White House ............................................................................................. 333
14.11 A basic map of the North Carolina congressional districts .................................................................................... 338
14.12 Bichromatic choropleth map of the results of the 2012 congressional elections in North Carolina 341
14.13 Full color choropleth of the results of the 2012 congressional elections in North Carolina .................. 342
14.14 Static image from a leaflet plot of the North Carolina congressional districts .................................................. 344
14.15 Airports served by Delta Airlines in 2006 ............................................................................................................. 347
14.16 Full route map for Delta Airlines in 2006 ............................................................................................................. 348
14.17 Static image from a leaflet plot of the historical Delta airlines route map ...................................................... 350
14.18 U.S ........................................................................................................................................................................... 351
14.19 U.S ......................................................................................................................................................................... 352
14.20 Screenshot of the North Carolina congressional districts as rendered in Google Earth, after exporting to KML. Compare with Figure 14.13 ........................................................................ 354

15.1 Speaking parts in *Macbeth* for four major characters ........................................................................... 361
15.2 A word cloud of terms that appear in the abstracts of arXiv papers on data science 366
### LIST OF FIGURES

- **15.3** Distribution of the number of characters in a sample of tweets  
  - Page 371
- **15.4** Distribution of the number of retweets in a sample of tweets  
  - Page 372
- **16.1** Two Erdős–Rényi random graphs on 100 vertices with different values of $p$  
  - Page 379
- **16.2** Simulation of connectedness of ER random graphs on 1,000 vertices  
  - Page 380
- **16.3** Degree distribution for two random graphs  
  - Page 381
- **16.4** Visualization of Hollywood network for popular 2012 movies  
  - Page 385
- **16.5** Distribution of degrees for actors in the Hollywood network of popular 2012 movies.  
  - Page 387
- **16.6** The Hollywood network for popular 2012 movies, in ggplot2  
  - Page 388
- **16.7** Atlantic 10 Conference network, NCAA men’s basketball, 1995–1996  
  - Page 396
- **B.1** Sample session in R  
  - Page 423
- **B.2** Documentation on the `mean()` function  
  - Page 425
- **C.1** Illustration of the location of the critical value for a 95% confidence interval for a mean  
  - Page 444
- **C.2** Cauchy distribution (solid line) and t-distribution with 4 degrees of freedom (dashed line).  
  - Page 447
- **C.3** Running average for t-distribution with four degrees of freedom and a Cauchy random variable (equivalent to a t-distribution with one degree of freedom). Note that while the former converges, the latter does not.  
  - Page 448
- **D.1** Generating a new R Markdown file in RStudio  
  - Page 457
- **D.2** Sample R Markdown input file  
  - Page 458
- **D.3** Formatted output from R Markdown example  
  - Page 460
- **E.1** Scatterplot of number of trail crossings as a function of highest daily temperature (in degrees Fahrenheit).  
  - Page 467
- **E.2** At left, the model based on the overall average high temperature  
  - Page 468
- **E.3** Visualization of parallel slopes model for the rail trail data  
  - Page 471
- **E.4** Visualization of interaction model for the rail trail data  
  - Page 473
- **E.5** Scatterplot of height as a function of age with superimposed linear model (blue) and smoother (green)  
  - Page 474
- **E.6** Scatterplot of volume as a function of high temperature with superimposed linear and smooth models for the rail trail data  
  - Page 475
- **E.7** Assessing linearity using a scatterplot of residuals versus fitted (predicted) values  
  - Page 476
- **E.8** Assessing normality assumption using a Q–Q plot  
  - Page 477
- **E.9** Assessing equal variance using a scale–location plot  
  - Page 478
- **E.10** Cook’s distance for rail trail model  
  - Page 479
- **E.11** Scatterplot of diabetes as a function of age with superimposed smoother  
  - Page 480
- **E.12** Scatterplot of diabetes as a function of BMI with superimposed smoother.  
  - Page 480
- **E.13** Predicted probabilities for diabetes as a function of BMI and age  
  - Page 481
- **F.1** Schematic of SQL-related R packages and their dependencies  
  - Page 493
Chapter 4

Data wrangling

This chapter introduces basics of how to wrangle data in R. Wrangling skills will provide an intellectual and practical foundation for working with modern data.

4.1 A grammar for data wrangling

In much the same way that ggplot2 presents a grammar for data graphics, the dplyr package presents a grammar for data wrangling [234]. Hadley Wickham, one of the authors of dplyr, has identified five verbs for working with data in a data frame:

- **select()** take a subset of the columns (i.e., features, variables)
- **filter()** take a subset of the rows (i.e., observations)
- **mutate()** add or modify existing columns
- **arrange()** sort the rows
- **summarize()** aggregate the data across rows (e.g., group it according to some criteria)

Each of these functions takes a data frame as its first argument, and returns a data frame. Thus, these five verbs can be used in conjunction with each other to provide a powerful means to slice-and-dice a single table of data. As with any grammar, what these verbs mean on their own is one thing, but being able to combine these verbs with nouns (i.e., data frames) creates an infinite space for data wrangling. Mastery of these five verbs can make the computation of most any descriptive statistic a breeze and facilitate further analysis. Wickham’s approach is inspired by his desire to blur the boundaries between R and the ubiquitous relational database querying syntax SQL. When we revisit SQL in Chapter 12, we will see the close relationship between these two computing paradigms. A related concept more popular in business settings is the OLAP (online analytical processing) hypercube, which refers to the process by which multidimensional data is “sliced-and-diced.”

4.1.1 **select() and filter()**

The two simplest of the five verbs are **filter()** and **select()**, which allow you to return only a subset of the rows or columns of a data frame, respectively. Generally, if we have a data frame that consists of \( n \) rows and \( p \) columns, Figures 4.1 and 4.2 illustrate the effect of filtering this data frame based on a condition on one of the columns, and selecting a subset of the columns, respectively.
CHAPTER 4. DATA WRANGLING

Figure 4.1: The \texttt{filter()} function. At left, a data frame that contains matching entries in a certain column for only a subset of the rows. At right, the resulting data frame after filtering.

Figure 4.2: The \texttt{select()} function. At left, a data frame, from which we retrieve only a few of the columns. At right, the resulting data frame after selecting those columns.

Specifically, we will demonstrate the use of these functions on the \texttt{presidential} data frame (from the \texttt{ggplot2} package), which contains \( p = 4 \) variables about the terms of \( n = 11 \) recent U.S. Presidents.

\begin{verbatim}
library(mdsr)
presidential

# A tibble: 11 x 4
   name     start     end party
   <chr>     <date>    <date> <chr>
1 Eisenhower 1953-01-20 1961-01-20 Republican
2 Kennedy    1961-01-20 1963-11-22 Democratic
3 Johnson    1963-11-22 1969-01-20 Democratic
4 Nixon      1969-01-20 1974-08-09 Republican
5 Ford       1974-08-09 1977-01-20 Republican
6 Carter     1977-01-20 1981-01-20 Democratic
7 Reagan     1981-01-20 1989-01-20 Republican
8 Bush       1989-01-20 1993-01-20 Republican
9 Clinton    1993-01-20 2001-01-20 Democratic
10 Bush       2001-01-20 2009-01-20 Republican
11 Obama      2009-01-20 2017-01-20 Democratic
\end{verbatim}

To retrieve only the names and party affiliations of these presidents, we would use \texttt{select()}. The first argument to the \texttt{select()} function is the data frame, followed by an arbitrarily long list of column names, separated by commas. Note that it is not necessary to wrap the column names in quotation marks.
select(presidential, name, party)

# A tibble: 11 2
  name    party
  <chr>   <chr>
1 Eisenhower Republican
2 Kennedy Democratic
3 Johnson Democratic
4 Nixon Republican
5 Ford Republican
6 Carter Democratic
7 Reagan Republican
8 Bush Republican
9 Clinton Democratic
10 Bush Republican
11 Obama Democratic

Similarly, the first argument to `filter()` is a data frame, and subsequent arguments are logical conditions that are evaluated on any involved columns. Thus, if we want to retrieve only those rows that pertain to Republican presidents, we need to specify that the value of the `party` variable is equal to `Republican`.

filter(presidential, party == "Republican")

# A tibble: 6 4
  name     start       end   party
  <chr>    <date>       <date> <chr>
1 Eisenhower 1953-01-20 1961-01-20 Republican
2 Nixon 1969-01-20 1974-08-09 Republican
3 Ford 1974-08-09 1977-01-20 Republican
4 Reagan 1981-01-20 1989-01-20 Republican
5 Bush 1989-01-20 1993-01-20 Republican
6 Bush 2001-01-20 2009-01-20 Republican

Note that the `==` is a test for equality. If we were to use only a single equal sign here, we would be asserting that the value of `party` was `Republican`. This would cause all of the rows of `presidential` to be returned, since we would have overwritten the actual values of the `party` variable. Note also the quotation marks around `Republican` are necessary here, since `Republican` is a literal value, and not a variable name.

Naturally, combining the `filter()` and `select()` commands enables one to drill down to very specific pieces of information. For example, we can find which Democratic presidents served since Watergate.

select(filter(presidential, start > "1973-01-01" & party == "Democratic"), name)

# A tibble: 3 1
  name
  <chr>
1 Carter
2 Clinton
3 Obama
CHAPTER 4. DATA WRANGLING

Figure 4.3: The `mutate()` function. At left, a data frame. At right, the resulting data frame after adding a new column.

In the syntax demonstrated above, the `filter()` operation is nested inside the `select()` operation. As noted above, each of the five verbs takes and returns a data frame, which makes this type of nesting possible. Shortly, we will see how these verbs can be chained together to make rather long expressions that can become very difficult to read. Instead, we recommend the use of the `%>%` (pipe) operator. Pipe-forwarding is an alternative to nesting that yields code that can be easily read from top to bottom. With the pipe, we can write the same expression as above in this more readable syntax.

```r
presidential %>%
  filter(start > 1973 & party == "Democratic") %>%
  select(name)
```

# A tibble: 3 1
name <chr>
1 Carter
2 Clinton
3 Obama

This expression is called a pipeline. Notice how the expression

```r
dataframe %>% filter(condition)
```

is equivalent to `filter(dataframe, condition)`. In later examples we will see how this operator can make our code more readable and efficient, particularly for complex operations on large data sets.

4.1.2 `mutate()` and `rename()`

Frequently, in the process of conducting our analysis, we will create, re-define, and rename some of our variables. The functions `mutate()` and `rename()` provide these capabilities. A graphical illustration of the `mutate()` operation is shown in Figure 4.3.

While we have the raw data on when each of these presidents took and relinquished office, we don’t actually have a numeric variable giving the length of each president’s term. Of course, we can derive this information from the dates given, and add the result as a new column to our data frame. This date arithmetic is made easier through the use of the `lubridate` package, which we use to compute the number of exact years (`eyears(1)`) that elapsed since during the `interval()` from the `start` until the end of each president’s term.
4.1. A GRAMMAR FOR DATA WRANGLING

In this situation, it is generally considered good style to create a new object rather than clobbering the one that comes from an external source. To preserve the existing presidential data frame, we save the result of `mutate()` as a new object called `mypresidents`.

```r
library(lubridate)

mypresidents <- presidential %>%
    mutate(term.length = interval(start, end) / eyears(1))

mypresidents
# A tibble: 11
      name start          end   party term.length
   <chr> <date>          <date> <chr>        <dbl>
1  Eisenhower 1953-01-20 1961-01-20 Republican   8.01
2    Kennedy 1961-01-20 1963-11-22 Democratic     2.84
3  Johnson 1963-11-22 1969-01-20 Democratic      5.17
4    Nixon 1969-01-20 1974-08-09 Republican      5.55
5     Ford 1974-08-09 1977-01-20 Republican       2.45
6    Carter 1977-01-20 1981-01-20 Democratic      4.00
7   Reagan 1981-01-20 1989-01-20 Republican      8.01
8     Bush 1989-01-20 1993-01-20 Republican       4.00
9  Clinton 1993-01-20 2001-01-20 Democratic      8.01
10    Bush 2001-01-20 2009-01-20 Republican       8.01
11   Obama 2009-01-20 2017-01-20 Democratic      8.01
```

The `mutate()` function can also be used to modify the data in an existing column. Suppose that we wanted to add to our data frame a variable containing the year in which each president was elected. Our first naive attempt is to assume that every president was elected in the year before he took office. Note that `mutate()` returns a data frame, so if we want to modify our existing data frame, we need to overwrite it with the results.

```r
mypresidents <- mypresidents %>%
    mutate(elected = year(start) - 1)

mypresidents
# A tibble: 11
      name start          end   party term.length elected
   <chr> <date>          <date> <chr>        <dbl>    <dbl>
1  Eisenhower 1953-01-20 1961-01-20 Republican   8.01      1952
2    Kennedy 1961-01-20 1963-11-22 Democratic     2.84      1960
4    Nixon 1969-01-20 1974-08-09 Republican      5.55      1968
5     Ford 1974-08-09 1977-01-20 Republican       2.45      1973
6    Carter 1977-01-20 1981-01-20 Democratic      4.00      1976
8     Bush 1989-01-20 1993-01-20 Republican       4.00      1988
9  Clinton 1993-01-20 2001-01-20 Democratic      8.01      1992
10    Bush 2001-01-20 2009-01-20 Republican       8.01      2000
11   Obama 2009-01-20 2017-01-20 Democratic      8.01      2008
```

Some aspects of this data set are wrong, because presidential elections are only held every four years. Lyndon Johnson assumed the office after President Kennedy was assassinated in 1963, and Gerald Ford took over after President Nixon resigned in 1974. Thus, there were no presidential elections in 1962 or 1973, as suggested in our data frame. We should overwrite...
these values with NA’s—which is how R denotes missing values. We can use the `ifelse()` function to do this. Here, if the value of `elected` is either 1962 or 1973, we overwrite that value with NA.\(^1\) Otherwise, we overwrite it with the same value that it currently has. In this case, instead of checking to see whether the value of `elected` equals 1962 or 1973, for brevity we can use the `%in%` operator to check to see whether the value of `elected` belongs to the vector consisting of 1962 and 1973.

```r
mypresidents <- mypresidents %>% 
mutate(elected = ifelse((elected %in% c(1962, 1973)), NA, elected))
mypresidents
```

Finally, it is considered bad practice to use periods in the name of functions, data frames, and variables in R. Ill-advised periods could conflict with R’s use of generic functions (i.e., R’s mechanism for method overloading). Thus, we should change the name of the `term.length` column that we created earlier. In this book, we will use snake case for function and variable names. We can achieve this using the `rename()` function.

```r
Pro Tip: Don’t use periods in the names of functions, data frames, or variables, as this can conflict with R’s programming model.
```

```r
mypresidents <- mypresidents %>% rename(term_length = term.length)
mypresidents
```

---

^1Incidentally, Johnson was elected in 1964 as an incumbent.
4.1. A GRAMMAR FOR DATA WRANGLING

Figure 4.4: The `arrange()` function. At left, a data frame with an ordinal variable. At right, the resulting data frame after sorting the rows in descending order of that variable.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Clinton</td>
<td>1993-01-20</td>
<td>2001-01-20</td>
<td>Democratic</td>
<td>8.01</td>
</tr>
<tr>
<td>10</td>
<td>Bush</td>
<td>2001-01-20</td>
<td>2009-01-20</td>
<td>Republican</td>
<td>8.01</td>
</tr>
<tr>
<td>11</td>
<td>Obama</td>
<td>2009-01-20</td>
<td>2017-01-20</td>
<td>Democratic</td>
<td>8.01</td>
</tr>
</tbody>
</table>

4.1.3 `arrange()`

The function `sort()` will sort a vector, but not a data frame. The function that will sort a data frame is called `arrange()`, and its behavior is illustrated in Figure 4.4.

In order to use `arrange()` on a data frame, you have to specify the data frame, and the column by which you want it to be sorted. You also have to specify the direction in which you want it to be sorted. Specifying multiple sort conditions will result in any ties being broken. Thus, to sort our presidential data frame by the length of each president’s term, we specify that we want the column `term_length` in descending order.

```
mypresidents %>% arrange(desc(term_length))
```

# A tibble: 11 × 6

<table>
<thead>
<tr>
<th>name</th>
<th>start</th>
<th>end</th>
<th>party</th>
<th>term_length</th>
<th>elected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenhower</td>
<td>1953-01-20</td>
<td>1961-01-20</td>
<td>Republican</td>
<td>8.01</td>
<td>1952</td>
</tr>
<tr>
<td>Clinton</td>
<td>1993-01-20</td>
<td>2001-01-20</td>
<td>Democratic</td>
<td>8.01</td>
<td>1992</td>
</tr>
<tr>
<td>Bush</td>
<td>2001-01-20</td>
<td>2009-01-20</td>
<td>Republican</td>
<td>8.01</td>
<td>2000</td>
</tr>
<tr>
<td>Obama</td>
<td>2009-01-20</td>
<td>2017-01-20</td>
<td>Democratic</td>
<td>8.01</td>
<td>2008</td>
</tr>
<tr>
<td>Nixon</td>
<td>1969-01-20</td>
<td>1974-08-09</td>
<td>Republican</td>
<td>5.55</td>
<td>1968</td>
</tr>
<tr>
<td>Johnson</td>
<td>1963-11-22</td>
<td>1969-01-20</td>
<td>Democratic</td>
<td>5.17</td>
<td>NA</td>
</tr>
<tr>
<td>Carter</td>
<td>1977-01-20</td>
<td>1981-01-20</td>
<td>Democratic</td>
<td>4.00</td>
<td>1976</td>
</tr>
<tr>
<td>Bush</td>
<td>1989-01-20</td>
<td>1993-01-20</td>
<td>Republican</td>
<td>4.00</td>
<td>1988</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1961-01-20</td>
<td>1963-11-22</td>
<td>Democratic</td>
<td>2.84</td>
<td>1960</td>
</tr>
<tr>
<td>Ford</td>
<td>1974-08-09</td>
<td>1977-01-20</td>
<td>Republican</td>
<td>2.45</td>
<td>NA</td>
</tr>
</tbody>
</table>

A number of presidents completed either one or two full terms, and thus have the exact same term length (4 or 8 years, respectively). To break these ties, we can further sort by party and elected.

```
mypresidents %>% arrange(desc(term_length), party, elected)
```

# A tibble: 11 × 6
CHAPTER 4. DATA WRANGLING

Figure 4.5: The `summarize()` function. At left, a data frame. At right, the resulting data frame after aggregating three of the columns.

<table>
<thead>
<tr>
<th>name</th>
<th>start</th>
<th>end</th>
<th>party</th>
<th>term_length</th>
<th>elected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clinton</td>
<td>1993-01-20</td>
<td>2001-01-20</td>
<td>Democratic</td>
<td>8.01</td>
<td>1992</td>
</tr>
<tr>
<td>Obama</td>
<td>2009-01-20</td>
<td>2017-01-20</td>
<td>Democratic</td>
<td>8.01</td>
<td>2008</td>
</tr>
<tr>
<td>Eisenhower</td>
<td>1953-01-20</td>
<td>1961-01-20</td>
<td>Republican</td>
<td>8.01</td>
<td>1952</td>
</tr>
<tr>
<td>Bush</td>
<td>2001-01-20</td>
<td>2008-01-20</td>
<td>Republican</td>
<td>8.01</td>
<td>2000</td>
</tr>
<tr>
<td>Nixon</td>
<td>1969-01-20</td>
<td>1974-01-09</td>
<td>Republican</td>
<td>5.55</td>
<td>1968</td>
</tr>
<tr>
<td>Johnson</td>
<td>1963-11-22</td>
<td>1969-01-20</td>
<td>Democratic</td>
<td>5.17</td>
<td>NA</td>
</tr>
<tr>
<td>Carter</td>
<td>1977-01-20</td>
<td>1981-01-20</td>
<td>Democratic</td>
<td>4.00</td>
<td>1976</td>
</tr>
<tr>
<td>Bush</td>
<td>1989-01-20</td>
<td>1993-01-20</td>
<td>Republican</td>
<td>4.00</td>
<td>1988</td>
</tr>
<tr>
<td>Kennedy</td>
<td>1961-01-20</td>
<td>1963-11-22</td>
<td>Democratic</td>
<td>2.84</td>
<td>1960</td>
</tr>
<tr>
<td>Ford</td>
<td>1974-08-09</td>
<td>1977-01-20</td>
<td>Republican</td>
<td>2.45</td>
<td>NA</td>
</tr>
</tbody>
</table>

Note that the default sort order is ascending order, so we do not need to specify an order if that is what we want.

### 4.1.4 `summarize()` with `group_by()`

Our last of the five verbs for single-table analysis is `summarize()`, which is nearly always used in conjunction with `group_by()`. The previous four verbs provided us with means to manipulate a data frame in powerful and flexible ways. But the extent of the analysis we can perform with these four verbs alone is limited. On the other hand, `summarize()` with `group_by()` enables us to make comparisons.

When used alone, `summarize()` collapses a data frame into a single row. This is illustrated in Figure 4.5. Critically, we have to specify how we want to reduce an entire column of data into a single value. The method of aggregation that we specify controls what will appear in the output.

```r
mypresidents %>%
  summarize(
    N = n(), first_year = min(year(start)), last_year = max(year(end)),
    num_dems = sum(party == "Democratic"),
    years = sum(term_length),
    avg_term_length = mean(term_length))
```

# A tibble: 1 x 6
4.1. A GRAMMAR FOR DATA WRANGLING

The first argument to `summarize()` is a data frame, followed by a list of variables that will appear in the output. Note that every variable in the output is defined by operations performed on vectors—not on individual values. This is essential, since if the specification of an output variable is not an operation on a vector, there is no way for R to know how to collapse each column.

In this example, the function `n()` simply counts the number of rows. This is almost always useful information.

**Pro Tip:** To help ensure that data aggregation is being done correctly, use `n()` every time you use `summarize()`.

The next two variables determine the first year that one of these presidents assumed office. This is the smallest year in the `start` column. Similarly, the most recent year is the largest year in the `end` column. The variable `num_dems` simply counts the number of rows in which the value of the `party` variable was Democratic. Finally, the last two variables compute the sum and average of the `term_length` variable. Thus, we can quickly see that 5 of the 11 presidents who served from 1953 to 2017 were Democrats, and the average term length over these 64 years was about 5.8 years.

This begs the question of whether Democratic or Republican presidents served a longer average term during this time period. To figure this out, we can just execute `summarize()` again, but this time, instead of the first argument being the data frame `mypresidents`, we will specify that the rows of the `mypresidents` data frame should be grouped by the values of the `party` variable. In this manner, the same computations as above will be carried out for each party separately.

```
mypresidents %>%
group_by(party) %>%
summarize(
  N = n(), first_year = min(year(start)), last_year = max(year(end)),
  num_dems = sum(party == "Democratic"),
  years = sum(term_length),
  avg_term_length = mean(term_length))
```

This provides us with the valuable information that the six Republican presidents served an average of 6 years in office, while the five Democratic presidents served an average of only 5.6. As with all of the `dplyr` verbs, the final output is a data frame.

**Pro Tip:** In this chapter we are using the `dplyr` package. The most common way to extract data from data tables is with SQL (structured query language). We’ll introduce SQL in Chapter 12. The `dplyr` package provides a new interface that fits more smoothly into an overall data analysis workflow and is, in our opinion, easier to learn. Once you...
understand data wrangling with dplyr, it’s straightforward to learn SQL if needed. And dplyr can work as an interface to many systems that use SQL internally.

4.2 Extended example: Ben’s time with the Mets

In this extended example, we will continue to explore Sean Lahman’s historical baseball database, which contains complete seasonal records for all players on all Major League Baseball teams going back to 1871. These data are made available in R via the Lahman package. Here again, while domain knowledge may be helpful, it is not necessary to follow the example. To flesh out your understanding, try reading the Wikipedia entry on Major League Baseball.

```r
library(Lahman)
dim(Teams)
```

The `Teams` table contains the seasonal results of every major league team in every season since 1871. There are 2805 rows and 48 columns in this table, which is far too much to show here, and would make for a quite unwieldy spreadsheet. Of course, we can take a peek at what this table looks like by printing the first few rows of the table to the screen with the `head()` command, but we won’t print that on the page of this book.

Ben worked for the New York Mets from 2004 to 2012. How did the team do during those years? We can use `filter()` and `select()` to quickly identify only those pieces of information that we care about.

```r
mets <- Teams %>% filter(teamID == "NYN")
myMets <- mets %>% filter(yearID %in% 2004:2012)
myMets %>% select(yearID, teamID, W, L)
```

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
</tr>
<tr>
<td>2005</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>2006</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
</tr>
<tr>
<td>2007</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
</tr>
<tr>
<td>2008</td>
<td>NYN</td>
<td>89</td>
<td>73</td>
</tr>
<tr>
<td>2009</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
</tr>
<tr>
<td>2010</td>
<td>NYN</td>
<td>79</td>
<td>83</td>
</tr>
<tr>
<td>2011</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
</tr>
<tr>
<td>2012</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
</tr>
</tbody>
</table>

Notice that we have broken this down into three steps. First, we filter the rows of the `Teams` data frame into only those teams that correspond to the New York Mets. There are 54 of those, since the Mets joined the National League in 1962.

```r
nrow(mets)
```

[1] 54

---

2 The `teamID` value of NYN stands for the New York National League club.
4.2. EXTENDED EXAMPLE: BEN’S TIME WITH THE METS

Next, we filtered these data so as to include only those seasons in which Ben worked for the team—those with yearID between 2004 and 2012. Finally, we printed to the screen only those columns that were relevant to our question: the year, the team’s ID, and the number of wins and losses that the team had.

While this process is logical, the code can get unruly, since two ancillary data frames (mets and myMets) were created during the process. It may be the case that we’d like to use data frames later in the analysis. But if not, they are just cluttering our workspace, and eating up memory. A more streamlined way to achieve the same result would be to nest these commands together.

```
select(filter(mets, teamID == "NYN" & yearID %in% 2004:2012),
yearID, teamID, W, L)
```

This way, no additional data frames were created. However, it is easy to see that as we nest more and more of these operations together, this code could become difficult to read. To maintain readability, we instead chain these operations, rather than nest them (and get the same exact results).

```
Teams %>%
  select(yearID, teamID, W, L) %>%
  filter(teamID == "NYN" & yearID %in% 2004:2012)
```

This piping syntax (introduced in Section 4.1.1) is provided by the `dplyr` package. It retains the step-by-step logic of our original code, while being easily readable, and efficient with respect to memory and the creation of temporary data frames. In fact, there are also performance enhancements under the hood that make this the most efficient way to do these kinds of computations. For these reasons we will use this syntax whenever possible throughout the book. Note that we only have to type `Teams` once—it is implied by the pipe operator `%>%` that the subsequent command takes the previous data frame as its first argument. Thus, `df %>% f(y)` is equivalent to `f(df, y).

We’ve answered the simple question of how the Mets performed during the time that Ben was there, but since we are data scientists, we are interested in deeper questions. For example, some of these seasons were subpar—the Mets had more losses than wins. Did the team just get unlucky in those seasons? Or did they actually play as badly as their record indicates?

In order to answer this question, we need a model for expected winning percentage. It turns out that one of the most widely used contributions to the field of baseball analytics (courtesy of Bill James) is exactly that. This model translates the number of runs

---

3In baseball, a team scores a run when a player traverses the bases and return to home plate. The team with the most runs in each game wins, and no ties are allowed.
a team scores and allows over the course of an entire season into an expectation for how many games they should have won. The simplest version of this model is this:

\[
\hat{WPct} = \frac{1}{1 + (\frac{RA}{RS})^2},
\]

where \( RA \) is the number of runs the team allows, \( RS \) is the number of runs that the team scores, and \( \hat{WPct} \) is the team’s expected winning percentage. Luckily for us, the runs scored and allowed are present in the Teams table, so let’s grab them and save them in a new data frame.

```r
metsBen <- Teams %>% select(yearID, teamID, W, L, R, RA) %>%
  filter(teamID == "NYN" & yearID %in% 2004:2012)
```

First, note that the runs-scored variable is called \( R \) in the Teams table, but to stick with our notation we want to rename it \( RS \).

```r
metsBen <- metsBen %>% rename(RS = R)
```

Next, we need to compute the team’s actual winning percentage in each of these seasons. Thus, we need to add a new column to our data frame, and we do this with the `mutate()` command.

```r
metsBen <- metsBen %>% mutate(WPct = W / (W + L))
```
4.2. EXTENDED EXAMPLE: BEN’S TIME WITH THE METS

We also need to compute the model estimates for winning percentage.

```r
metsBen <- metsBen %>% mutate(WPct_hat = 1 / (1 + (RA/RS)^2))
metsBen
```

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
<td>684</td>
<td>731</td>
<td>0.438</td>
<td>0.467</td>
</tr>
<tr>
<td>2005</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>648</td>
<td>0.512</td>
<td>0.554</td>
</tr>
<tr>
<td>2006</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
<td>834</td>
<td>731</td>
<td>0.599</td>
<td>0.566</td>
</tr>
<tr>
<td>2007</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
<td>804</td>
<td>750</td>
<td>0.543</td>
<td>0.535</td>
</tr>
<tr>
<td>2008</td>
<td>NYN</td>
<td>89</td>
<td>73</td>
<td>799</td>
<td>715</td>
<td>0.549</td>
<td>0.555</td>
</tr>
<tr>
<td>2009</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
</tr>
<tr>
<td>2010</td>
<td>NYN</td>
<td>79</td>
<td>83</td>
<td>656</td>
<td>652</td>
<td>0.488</td>
<td>0.503</td>
</tr>
<tr>
<td>2011</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
<td>718</td>
<td>742</td>
<td>0.475</td>
<td>0.484</td>
</tr>
<tr>
<td>2012</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
<td>650</td>
<td>709</td>
<td>0.457</td>
<td>0.457</td>
</tr>
</tbody>
</table>

The expected number of wins is then equal to the product of the expected winning percentage times the number of games.

```r
metsBen <- metsBen %>% mutate(W_hat = WPct_hat * (W + L))
metsBen
```

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
<th>W_hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
<td>684</td>
<td>731</td>
<td>0.438</td>
<td>0.467</td>
<td>75.6</td>
</tr>
<tr>
<td>2005</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>648</td>
<td>0.512</td>
<td>0.554</td>
<td>89.7</td>
</tr>
<tr>
<td>2006</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
<td>834</td>
<td>731</td>
<td>0.599</td>
<td>0.566</td>
<td>91.6</td>
</tr>
<tr>
<td>2007</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
<td>804</td>
<td>750</td>
<td>0.543</td>
<td>0.535</td>
<td>86.6</td>
</tr>
<tr>
<td>2008</td>
<td>NYN</td>
<td>89</td>
<td>73</td>
<td>799</td>
<td>715</td>
<td>0.549</td>
<td>0.555</td>
<td>90.0</td>
</tr>
<tr>
<td>2009</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
<td>71.3</td>
</tr>
<tr>
<td>2010</td>
<td>NYN</td>
<td>79</td>
<td>83</td>
<td>656</td>
<td>652</td>
<td>0.488</td>
<td>0.503</td>
<td>81.5</td>
</tr>
<tr>
<td>2011</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
<td>718</td>
<td>742</td>
<td>0.475</td>
<td>0.484</td>
<td>78.3</td>
</tr>
<tr>
<td>2012</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
<td>650</td>
<td>709</td>
<td>0.457</td>
<td>0.457</td>
<td>74.0</td>
</tr>
</tbody>
</table>

In this case, the Mets’ fortunes were better than expected in three of these seasons, and worse than expected in the other six.

```r
filter(metsBen, W >= W_hat)
```

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
<th>W_hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
<td>834</td>
<td>731</td>
<td>0.599</td>
<td>0.566</td>
<td>91.6</td>
</tr>
<tr>
<td>2007</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
<td>804</td>
<td>750</td>
<td>0.543</td>
<td>0.535</td>
<td>86.6</td>
</tr>
<tr>
<td>2012</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
<td>650</td>
<td>709</td>
<td>0.457</td>
<td>0.457</td>
<td>74.0</td>
</tr>
</tbody>
</table>
filter(metsBen, W < W_hat)

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
<th>W_hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
<td>684</td>
<td>731</td>
<td>0.438</td>
<td>0.467</td>
<td>75.6</td>
</tr>
<tr>
<td>2</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>648</td>
<td>0.512</td>
<td>0.554</td>
<td>89.7</td>
</tr>
<tr>
<td>3</td>
<td>NYN</td>
<td>89</td>
<td>73</td>
<td>799</td>
<td>715</td>
<td>0.549</td>
<td>0.555</td>
<td>90.0</td>
</tr>
<tr>
<td>4</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
<td>71.3</td>
</tr>
<tr>
<td>5</td>
<td>NYN</td>
<td>79</td>
<td>83</td>
<td>656</td>
<td>652</td>
<td>0.488</td>
<td>0.503</td>
<td>81.5</td>
</tr>
<tr>
<td>6</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
<td>718</td>
<td>742</td>
<td>0.475</td>
<td>0.484</td>
<td>78.3</td>
</tr>
</tbody>
</table>

Naturally, the Mets experienced ups and downs during Ben’s time with the team. Which seasons were best? To figure this out, we can simply sort the rows of the data frame.

arrange(metsBen, desc(WPct))

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
<th>W_hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
<td>834</td>
<td>731</td>
<td>0.599</td>
<td>0.566</td>
<td>91.6</td>
</tr>
<tr>
<td>2</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
<td>804</td>
<td>750</td>
<td>0.543</td>
<td>0.535</td>
<td>86.6</td>
</tr>
<tr>
<td>3</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>757</td>
<td>0.512</td>
<td>0.554</td>
<td>89.7</td>
</tr>
<tr>
<td>4</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
<td>718</td>
<td>742</td>
<td>0.475</td>
<td>0.484</td>
<td>78.3</td>
</tr>
<tr>
<td>5</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
<td>650</td>
<td>709</td>
<td>0.457</td>
<td>0.457</td>
<td>74.0</td>
</tr>
<tr>
<td>6</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
<td>684</td>
<td>731</td>
<td>0.438</td>
<td>0.467</td>
<td>75.6</td>
</tr>
<tr>
<td>7</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
<td>71.3</td>
</tr>
</tbody>
</table>

In 2006, the Mets had the best record in baseball during the regular season and nearly made the World Series. But how do these seasons rank in terms of the team’s performance relative to our model?

metsBen %>%
  mutate(Diff = W - W_hat) %>%
  arrange(desc(Diff))

<table>
<thead>
<tr>
<th>yearID</th>
<th>teamID</th>
<th>W</th>
<th>L</th>
<th>RS</th>
<th>RA</th>
<th>WPct</th>
<th>WPct_hat</th>
<th>W_hat</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYN</td>
<td>97</td>
<td>65</td>
<td>834</td>
<td>731</td>
<td>0.599</td>
<td>0.566</td>
<td>91.6</td>
<td>5.3840</td>
</tr>
<tr>
<td>2</td>
<td>NYN</td>
<td>88</td>
<td>74</td>
<td>804</td>
<td>750</td>
<td>0.543</td>
<td>0.535</td>
<td>86.6</td>
<td>1.3774</td>
</tr>
<tr>
<td>3</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>757</td>
<td>0.512</td>
<td>0.554</td>
<td>89.7</td>
<td>-0.9605</td>
</tr>
<tr>
<td>4</td>
<td>NYN</td>
<td>77</td>
<td>85</td>
<td>718</td>
<td>742</td>
<td>0.475</td>
<td>0.484</td>
<td>78.3</td>
<td>-2.4954</td>
</tr>
<tr>
<td>5</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
<td>71.3</td>
<td>-1.3377</td>
</tr>
<tr>
<td>6</td>
<td>NYN</td>
<td>74</td>
<td>88</td>
<td>650</td>
<td>709</td>
<td>0.457</td>
<td>0.457</td>
<td>74.0</td>
<td>0.0199</td>
</tr>
<tr>
<td>7</td>
<td>NYN</td>
<td>71</td>
<td>91</td>
<td>684</td>
<td>731</td>
<td>0.438</td>
<td>0.467</td>
<td>75.6</td>
<td>-4.6250</td>
</tr>
<tr>
<td>8</td>
<td>NYN</td>
<td>70</td>
<td>92</td>
<td>671</td>
<td>757</td>
<td>0.432</td>
<td>0.440</td>
<td>71.3</td>
<td>-6.7249</td>
</tr>
<tr>
<td>9</td>
<td>NYN</td>
<td>83</td>
<td>79</td>
<td>722</td>
<td>648</td>
<td>0.512</td>
<td>0.554</td>
<td>89.7</td>
<td>-6.7249</td>
</tr>
</tbody>
</table>

So 2006 was the Mets’ most fortunate year—since they won five more games than our model predicts—but 2005 was the least fortunate—since they won almost seven games fewer than our model predicts. This type of analysis helps us understand how the Mets performed in individual seasons, but we know that any randomness that occurs in individual years is likely to average out over time. So while it is clear that the Mets performed well in some seasons and poorly in others, what can we say about their overall performance?
4.2. EXTENDED EXAMPLE: BEN’S TIME WITH THE METS

We can easily summarize a single variable with the `favstats()` command from the `mosaic` package.

```r
favstats(~ W, data = metsBen)
```

```
  min  Q1 median  Q3  max    mean     sd  n  missing
70  74    79    88  97    80.9    9.1  9      0
```

This tells us that the Mets won nearly 81 games on average during Ben’s tenure, which corresponds almost exactly to a 0.500 winning percentage, since there are 162 games in a regular season. But we may be interested in aggregating more than one variable at a time. To do this, we use `summarize()`.

```r
metsBen %>%
  summarize(
    num_years = n(),
    total_W = sum(W), total_L = sum(L),
    total_WPct = sum(W) / sum(W + L),
    sum_resid = sum(W - W_hat)
  )
```

```
  num_years total_W total_L total_WPct    sum_resid
1         9    728    730     0.499   -10.6
```

In these nine years, the Mets had a combined record of 728 wins and 730 losses, for an overall winning percentage of .499. Just one extra win would have made them exactly 0.500! (If we could pick which game, we would definitely pick the final game of the 2007 season. A win there would have resulted in a playoff berth.) However, we’ve also learned that the team under-performed relative to our model by a total of 10.6 games over those nine seasons.

Usually, when we are summarizing a data frame like we did above, it is interesting to consider different groups. In this case, we can discretize these years into three chunks: one for each of the three general managers under whom Ben worked. Jim Duquette was the Mets’ general manager in 2004, Omar Minaya from 2005 to 2010, and Sandy Alderson from 2011 to 2012. We can define these eras using two nested `ifelse()` functions (the `case_when()` function in the `dplyr` package is helpful in such a setting).

```r
metsBen <- metsBen %>%
  mutate(
    gm = ifelse(yearID == 2004, "Duquette",
                ifelse(yearID >= 2011, "Alderson",
                        "Minaya")))
```

Next, we use the `gm` variable to define these groups with the `group_by()` operator. The combination of summarizing data by groups can be very powerful. Note that while the Mets were far more successful during Minaya’s regime (i.e., many more wins than losses), they did not meet expectations in any of the three periods.

```r
metsBen %>%
  group_by(gm) %>%
  summarize(
    num_years = n(),
    total_W = sum(W), total_L = sum(L),
    total_WPct = sum(W) / sum(W + L),
    sum_resid = sum(W - W_hat) %>%
    arrange(desc(sum_resid))
  )
```

```
# A tibble: 3  6
  gm     num_years total_W total_L total_WPct    sum_resid
1 Alderson  1     97     65    0.598    -16.1
2 Duquette  1     90     70    0.571    -19.3
3 Minaya   7     728    730     0.499   -10.6
```
The full power of the chaining operator is revealed below, where we do all the analysis at once, but retain the step-by-step logic.

```r
Teams %>%
  select(yearID, teamID, W, L, R, RA) %>%
  filter(teamID == "NYN" & yearID %in% 2004:2012) %>%
  rename(RS = R) %>%
  mutate(
    WPct = W / (W + L), WPct_hat = 1 / (1 + (RA/RS)^2),
    WHat = WPct_hat * (W + L),
    gm = ifelse(yearID == 2004, "Duquette",
                ifelse(yearID > 2011, "Alderson", "Minaya"))) %>%
  group_by(gm) %>%
  summarize(
    num_years = n(),
    total_W = sum(W),
    total_L = sum(L),
    total_WPct = sum(W) / sum(W + L),
    sum_resid = sum(W - WHat)) %>%
  arrange(desc(sum_resid))
```

Even more generally, we might be more interested in how the Mets performed relative to our model, in the context of all teams during that nine year period. All we need to do is remove the `teamID` filter and group by franchise (`franchID`) instead.

```r
Teams %>%
  select(yearID, teamID, franchID, W, L, R, RA) %>%
  filter(yearID %in% 2004:2012) %>%
  rename(RS = R) %>%
  mutate(
    WPct = W / (W + L), WPctHat = 1 / (1 + (RA/RS)^2),
    WHat = WPctHat * (W + L)) %>%
  group_by(franchID) %>%
  summarize(
    numYears = n(),
    totalW = sum(W),
    totalL = sum(L),
    totalWPct = sum(W) / sum(W + L),
    sumResid = sum(W - WHat)) %>%
  arrange(sumResid) %>%
  print(n = 6)
```

4.3 Combining multiple tables

In the previous section, we illustrated how the five verbs can be chained to perform operations on a single table. This single table is reminiscent of a single well-organized spreadsheet. But in the same way that a workbook can contain multiple spreadsheets, we will often work with multiple tables. In Chapter 12, we will describe how multiple tables related by unique identifiers called keys can be organized into a relational database management system.

It is more efficient for the computer to store and search tables in which “like is stored with like.” Thus, a database maintained by the Bureau of Transportation Statistics on the arrival times of U.S. commercial flights will consist of multiple tables, each of which contains data about different things. For example, the nycflights13 package contains one table about flights—each row in this table is a single flight. As there are many flights, you can imagine that this table will get very long—hundreds of thousands of rows per year. But there are other related kinds of information that we will want to know about these flights. We would certainly be interested in the particular airline to which each flight belonged. It would be inefficient to store the complete name of the airline (e.g., American Airlines Inc.) in every row of the flights table. A simple code (e.g., AA) would take up less space on disk. For small tables, the savings of storing two characters instead of 25 is insignificant, but for large tables, it can add up to noticeable savings both in terms of the size of data on disk, and the speed with which we can search it. However, we still want to have the full names of the airlines available if we need them. The solution is to store the data about airlines in a separate table called airlines, and to provide a key that links the data in the two tables together.

4.3.1 inner_join()

If we examine the first few rows of the flights table, we observe that the carrier column contains a two-character string corresponding to the airline.

```r
library(nycflights13)
head(flights, 3)
```

# A tibble: 3  19
  year month day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int> <int>         <int>     <dbl> <int>
1 2013   1   1     517          515       2   830
2 2013   1   1     533          529       4   850

---

4 Note that whereas the teamID that corresponds to the Mets is NYM, the value of the franchID variable is NYM.
In the `airlines` table, we have those same two-character strings, but also the full names of the airline.

```
head(airlines, 3)
# A tibble: 3
  carrier name
  <chr> <chr>
1 9E    Endeavor Air Inc.
2 AA    American Airlines Inc.
3 AS    Alaska Airlines Inc.
```

In order to retrieve a list of flights and the full names of the airlines that managed each flight, we need to match up the rows in the `flights` table with those rows in the `airlines` table that have the corresponding values for the `carrier` column in both tables. This is achieved with the function `inner_join()`:

```
flightsJoined <- flights %>%
  inner_join(airlines, by = c("carrier" = "carrier"))
glimpse(flightsJoined)
```

```
Observations: 336,776
Variables: 20
$ month <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ day <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ dep_time <int> 517, 533, 542, 544, 554, 554, 555, 556,...
$ sched_dep_time <int> 515, 533, 542, 544, 554, 554, 555, 556,...
$ dep_delay <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2,...
$ arr_time <int> 830, 850, 923, 1004, 812, 740, 913, 709,...
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723,...
$ arr_delay <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, -2,...
$ carrier <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV"...
$ flight <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708,...
$ tailnum <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN"...
$ origin <chr> "EWR", "LGA", "LGA", "LGA", "LGA", "EWR", "EWR"...
$ dest <chr> "IAH", "MIA", "BQN", "ATL", "ORD", "FLL"...
$ air_time <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229,...
$ distance <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
$ hour <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0,...
$ minute <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ time_hour <dttm> "2013-01-01 05:00:00", "2013-01-01 05:00:00", "2013"...
$ name <chr> "United Air Lines Inc.", "United Air Lines Inc....
```

Notice that the `flightsJoined` data frame now has an additional variable called `name`.
This is the column from airlines that is now attached to our combined data frame. Now we can view the full names of the airlines instead of the cryptic two-character codes.

```r
flightsJoined %>%
  select(carrier, name, flight, origin, dest) %>%
head(3)
```

# A tibble: 3 5
## carrier name flight origin dest
1 UA United Air Lines Inc. 1545 EWR IAH
2 UA United Air Lines Inc. 1714 LGA IAH
3 AA American Airlines Inc. 1141 JFK MIA

In an *inner join(*), the result set contains only those rows that have matches in both tables. In this case, all of the rows in `flights` have exactly one corresponding entry in `airlines`, so the number of rows in `flightsJoined` is the same as the number of rows in `flights` (this will not always be the case).

```r
nrow(flights)
[1] 336776
nrow(flightsJoined)
[1] 336776
```

**Pro Tip:** It is always a good idea to carefully check that the number of rows returned by a join operation is what you expected. In particular, you often want to check for rows in one table that matched to more than one row in the other table.

### 4.3.2 `left_join()`

Another commonly used type of join is a `left_join()`. Here the rows of the first table are *always* returned, regardless of whether there is a match in the second table.

Suppose that we are only interested in flights from the NYC airports to the West Coast. Specifically, we’re only interested in airports in the Pacific Time Zone. Thus, we filter the airports data frame to only include those 152 airports.

```r
airportsPT <- filter(airports, tz == -8)
nrow(airportsPT)
[1] 152
```

Now, if we perform an *inner join(* on `flights` and `airportsPT`, matching the destinations in `flights` to the FAA codes in `airports`, we retrieve only those flights that flew to our airports in the Pacific Time Zone.

```r
nycDestsPT <- flights %>% inner_join(airportsPT, by = c("dest" = "faa"))
nrow(nycDestsPT)
[1] 46324
```
However, if we use a `left_join()` with the same conditions, we retrieve all of the rows of `flights`. NA's are inserted into the columns where no matched data was found.

```r	nycDests <- flights %>% left_join(airportsPT, by = c("dest" = "faa"))
nrow(nycDests)
[1] 336776

sum(is.na(nycDests$name))
[1] 290452
```

Left joins are particularly useful in databases in which *referential integrity* is broken (not all of the keys are present—see Chapter 12).

### 4.4 Extended example: Manny Ramirez

In the context of baseball and the *Lahman* package, multiple tables are used to store information. The batting statistics of players are stored in one table (`Batting`), while information about people (most of whom are players) is in a different table (`Master`).

Every row in the `Batting` table contains the statistics accumulated by a single player during a single stint for a single team in a single year. Thus, a player like Manny Ramirez has many rows in the `Batting` table (21 in fact).

```r
manny <- filter(Batting, playerID == "ramirma02")
nrow(manny)
[1] 21
```

Using what we’ve learned, we can quickly tabulate Ramirez’s most common career offensive statistics. For those new to baseball, some additional background may be helpful. A hit (H) occurs when a batter reaches base safely. A home run (HR) occurs when the ball is hit out of the park or the runner advances through all of the bases during that play. Barry Bonds has the record for most home runs (762) hit in a career. A player’s batting average (BA) is the ratio of the number of hits to the number of eligible at-bats. The highest career batting average in major league baseball history of 0.366 was achieved by Ty Cobb—season averages above 0.300 are impressive. Finally, runs batted in (RBI) is the number of runners (including the batter in the case of a home run) that score during that batter’s at-bat. Hank Aaron has the record for most career RBIs with 2,297.

```r
manny %>% summarize(
  span = paste(min(yearID), max(yearID), sep = "-"),
  numYears = n_distinct(yearID), numTeams = n_distinct(teamID),
  BA = sum(H)/sum(AB), tH = sum(H), tHR = sum(HR), tRBI = sum(RBI))
```

<table>
<thead>
<tr>
<th>span</th>
<th>numYears</th>
<th>numTeams</th>
<th>BA</th>
<th>tH</th>
<th>tHR</th>
<th>tRBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993-2011</td>
<td>19</td>
<td>5</td>
<td>0.312</td>
<td>2574</td>
<td>555</td>
<td>1831</td>
</tr>
</tbody>
</table>

Notice how we have used the `paste()` function to combine results from multiple variables into a new variable, and how we have used the `n_distinct()` function to count the number of distinct rows. In his 19-year career, Ramirez hit 555 home runs, which puts him in the top 20 among all Major League players.
4.4. EXTENDED EXAMPLE: MANNY RAMIREZ

However, we also see that Ramirez played for five teams during his career. Did he perform equally well for each of them? Breaking his statistics down by team, or by league, is as easy as adding an appropriate group_by() command.

```r
manny %>%
group_by(teamID) %>%
summarize(
  span = paste(min(yearID), max(yearID), sep = "-")
  , numYears = n_distinct(yearID)
  , numTeams = n_distinct(teamID)
  , BA = sum(H)/sum(AB)
  , tH = sum(H)
  , tHR = sum(HR)
  , tRBI = sum(RBI))
arrange(span)
```

While Ramirez was very productive for Cleveland, Boston, and the Los Angeles Dodgers, his brief tours with the Chicago White Sox and Tampa Bay Rays were less than stellar. In the pipeline below, we can see that Ramirez spent the bulk of his career in the American League.

```r
manny %>%
group_by(lgID) %>%
summarize(
  span = paste(min(yearID), max(yearID), sep = "-")
  , numYears = n_distinct(yearID)
  , numTeams = n_distinct(teamID)
  , BA = sum(H)/sum(AB)
  , tH = sum(H)
  , tHR = sum(HR)
  , tRBI = sum(RBI))
arrange(span)
```

If Ramirez played in only 19 different seasons, why were there 21 rows attributed to him? Notice that in 2008, he was traded from the Boston Red Sox to the Los Angeles Dodgers, and thus played for both teams. Similarly, in 2010 he played for both the Dodgers and the Chicago White Sox. When summarizing data, it is critically important to understand exactly how the rows of your data frame are organized. To see what can go wrong here, suppose we were interested in tabulating the number of seasons in which Ramirez hit at least 30 home runs. The simplest solution is:

```r
manny %>%
filter(HR >= 30) %>%
nrow()
```

[1] 11
CHAPTER 4. DATA WRANGLING

But this answer is wrong, because in 2008, Ramirez hit 20 home runs for Boston before being traded and then 17 more for the Dodgers afterwards. Neither of those rows were counted, since they were both filtered out. Thus, the year 2008 does not appear among the 11 that we counted in the previous pipeline. Recall that each row in the `manny` data frame corresponds to one stint with one team in one year. On the other hand, the question asks us to consider each year, regardless of team. In order to get the right answer, we have to aggregate the rows by team. Thus, the correct solution is:

```r
manny %>%
group_by(yearID) %>%
summarize(tHR = sum(HR)) %>%
filter(tHR >= 30) %>%
nrow()
```

Note that the `filter()` operation is applied to `tHR`, the total number of home runs in a season, and not `HR`, the number of home runs in a single stint for a single team in a single season. (This distinction between filtering the rows of the original data versus the rows of the aggregated results will appear again in Chapter 12.)

We began this exercise by filtering the `Batting` table for the player with `playerID` equal to `ramirma02`. How did we know to use this identifier? This player ID is known as a key, and in fact, `playerID` is the primary key defined in the Master table. That is, every row in the Master table is uniquely identified by the value of `playerID`. Thus there is exactly one row in that table for which `playerID` is equal to `ramirma02`.

But how did we know that this ID corresponds to Manny Ramirez? We can search the Master table. The data in this table include characteristics about Manny Ramirez that do not change across multiple seasons (with the possible exception of his weight).

```r
Master %>% filter(nameLast == "Ramirez" & nameFirst == "Manny")
```

The `playerID` column forms a primary key in the Master table, but it does not in the Batting table, since as we saw previously, there were 21 rows with that `playerID`. In the Batting table, the `playerID` column is known as a foreign key, in that it references a primary key in another table. For our purposes, the presence of this column in both tables allows us to link them together. This way, we can combine data from the Batting table with data in the Master table. We do this with `inner_join()` by specifying the two tables that we want to join, and the corresponding columns in each table that provide the link. Thus, if we want to display Ramirez’s name in our previous result, as well as his age, we must join the Batting and Master tables together.
4.4. EXTENDED EXAMPLE: MANNY RAMIREZ

Batting `\%
filter(playerID == "ramirma02") %>%
inner_join(Master, by = c("playerID" = "playerID")) %>%
group_by(yearID) %>%
summarize(
    Age = max(yearID - birthYear),
    numTeams = n_distinct(teamID),
    BA = sum(H)/sum(AB),
    tH = sum(H),
    tHR = sum(HR),
    tRBI = sum(RBI)) %>%
arrange(yearID)

# A tibble: 19 7
  yearID Age numTeams  BA  tH  tHR tRBI
  <int> <int>    <int> <dbl> <int> <int> <int>
1  1993   21        1 0.1698   9   2    5
2  1994   22        1 0.2690  78  17   60
3  1995   23        1 0.3079 149  31  107
4  1996   24        1 0.3091 149  31  112
5  1997   25        1 0.3280 184  26   88
6  1998   26        1 0.2942 168  45  145
7  1999   27        1 0.3333 174  44  165
8  2000   28        1 0.3458 154  38  122
9  2001   29        1 0.3056 162  41  125
10 1992   30        1 0.3486 155  33  107
11 1993   31        1 0.3251 135  37  104
12 1994   32        1 0.3081 144  43  130
13 1995   33        1 0.2924 143  35  102
14 1996   34        1 0.3207 144  35  102
15 1997   35        1 0.2961 143  20  88
16 1998   36        1 0.3315 183  37  121
17 1999   37        1 0.2898 102  19  42
18 2000   38        1 0.2981  79  9   42
19 2001   39        1 0.0588  1  0   1

Pro Tip: Always specify the by argument that defines the join condition. Don’t rely on
the defaults.

Notice that even though Ramirez’s age is a constant for each season, we have to use a
vector operation (i.e., `max()`) in order to reduce any potential vector to a single number.

Which season was Ramirez’s best as a hitter? One relatively simple measurement of
batting prowess is OPS, or On-Base Plus Slugging Percentage, which is the simple sum
of two other statistics: On-Base Percentage (OBP) and Slugging Percentage (SLG). The
former basically measures the percentage of time that a batter reaches base safely, whether
it comes via a hit (H), a base on balls (BB), or from being hit by the pitch (HBP). The latter
measures the average number of bases advanced per at-bat (AB), where a single is worth
one base, a double (X2B) is worth two, a triple (X3B) is worth three, and a home run (HR)
is worth four. (Note that every hit is exactly one of a single, double, triple, or home run.)
Let’s add this statistic to our results and use it to rank the seasons.

mannyBySeason <- Batting `\%
filter(playerID == "ramirma02") %>%
inner_join(Master, by = c("playerID" = "playerID")) %>%
summarize(
    OPS = sum(H) + sum(BB) + sum(HBP)/sum(H) + sum(X2B)*2 + sum(X3B)*3 + sum(HR)*4)

# A tibble: 19 7
  yearID Age numTeams  BA  tH  tHR tRBI  OPS
  <int> <int>    <int> <dbl> <int> <int> <int> <dbl>
1  1993   21        1 0.1698   9   2    5 0.189
2  1994   22        1 0.2690  78  17   60 0.326
3  1995   23        1 0.3079 149  31  107 0.445
4  1996   24        1 0.3091 149  31  112 0.521
5  1997   25        1 0.3280 184  26   88 0.576
6  1998   26        1 0.2942 168  45  145 0.439
7  1999   27        1 0.3333 174  44  165 0.531
8  2000   28        1 0.3458 154  38  122 0.520
9  2001   29        1 0.3056 162  41  125 0.511
10 1992   30        1 0.3486 155  33  107 0.475
11 1993   31        1 0.3251 135  37  104 0.502
12 1994   32        1 0.3081 144  43  130 0.535
13 1995   33        1 0.2924 143  35  102 0.513
14 1996   34        1 0.3207 144  35  102 0.513
15 1997   35        1 0.2961 143  20  88 0.477
16 1998   36        1 0.3315 183  37  121 0.595
17 1999   37        1 0.2898 102  19  42 0.403
18 2000   38        1 0.2981  79  9   42 0.449
19 2001   39        1 0.0588  1  0   1 0.163
CHAPTER 4. DATA WRANGLING

```r
group_by(yearID) %>%
  summarize(
    Age = max(yearID - birthYear),
    numTeams = n_distinct(teamID),
    BA = sum(H) / sum(AB),
    tH = sum(H),
    tHR = sum(HR),
    tRBI = sum(RBI),
    OBP = sum(H + BB + HBP) / sum(AB + BB + SF + HBP),
    SLG = sum(H + X2B + 2*X3B + 3*HR) / sum(AB)) %>%
  mutate(OPS = OBP + SLG) %>%
  arrange(desc(OPS))
```

```r
mannyBySeason
# A tibble: 19 x 10
## yearID Age numTeams BA tH tHR tRBI OBP SLG OPS
## <int> <int> <int> <dbl> <int> <int> <int> <dbl> <dbl> <dbl>
1 2000 28 1 0.3508 154 38 122 0.4568 0.6970 1.154
2 1999 27 1 0.3333 174 44 165 0.4422 0.6628 1.105
3 2002 30 1 0.3486 152 33 107 0.4498 0.6468 1.097
4 2006 34 1 0.3207 144 35 102 0.4391 0.6192 1.058
5 2008 36 2 0.3315 183 37 121 0.4297 0.6014 1.031
6 2003 31 1 0.3251 185 37 104 0.4271 0.5870 1.014
7 2001 29 1 0.3062 162 41 125 0.4048 0.6087 1.014
8 2004 32 1 0.3033 175 43 130 0.3967 0.5870 1.014
9 2005 33 1 0.2924 162 45 144 0.3877 0.5939 0.982
10 1996 24 1 0.3091 130 33 132 0.3988 0.5818 0.981
11 1998 26 1 0.2942 158 45 145 0.3771 0.5789 0.976
12 1995 23 1 0.3079 146 31 107 0.4025 0.5797 0.960
13 1997 25 1 0.3280 184 26 88 0.4147 0.5383 0.953
14 2009 37 1 0.2898 102 19 63 0.4176 0.5312 0.949
15 2007 35 1 0.2961 143 20 88 0.3884 0.4928 0.881
16 1994 22 1 0.2690 78 17 60 0.3571 0.5207 0.878
17 2010 38 1 0.2981 79 9 42 0.4094 0.4604 0.870
18 1993 21 1 0.1698 9 2 5 0.2000 0.3019 0.502
19 2011 39 1 0.0588 1 0 1 0.0588 0.0588 0.118
```

We see that Ramirez’s OPS was highest in 2000. But 2000 was the height of the steroid era, when many sluggers were putting up tremendous offensive numbers. As data scientists, we know that it would be more instructive to put Ramirez’s OPS in context by comparing it to the league average OPS in each season—the resulting ratio is often called OPS+. To do this, we will need to compute those averages. Because there is missing data in some of these columns in some of these years, we need to invoke the `na.rm` argument to ignore that data.

```r
mlb <- Batting %>%
  filter(yearID %in% 1993:2011) %>%
  group_by(yearID) %>%
  summarize(lgOPS =
    sum(H + BB + HBP, na.rm = TRUE) / sum(AB + BB + SF + HBP, na.rm = TRUE) +
    sum(H + X2B + 2*X3B + 3*HR, na.rm = TRUE) / sum(AB, na.rm = TRUE))
```

Next, we need to match these league average OPS values to the corresponding entries for Ramirez. We can do this by joining these tables together, and computing the ratio of Ramirez’s OPS to that of the league average.
4.4. EXTENDED EXAMPLE: MANNY RAMIREZ

```r
mannyRatio <- mannyBySeason %>%
  inner_join(mlb, by = c("yearID" = "yearID")) %>%
  mutate(OPSplus = OPS / lgOPS) %>%
  select(yearID, Age, OPS, lgOPS, OPSplus) %>%
  arrange(desc(OPSplus))

mannyRatio
```

```
# A tibble: 19 x 5
  yearID Age  OPS lgOPS OPSplus
  <int> <int> <dbl> <dbl>    <dbl>
1    2000  28  1.154  0.782    1.475
2    2002  30  1.097  0.748    1.466
3    1999  27  1.105  0.778    1.420
4    2006  34  1.058  0.768    1.377
5    2008  36  1.031  0.749    1.376
6    2003  31  1.014  0.755    1.344
7    2001  29  1.014  0.759    1.336
8    2004  32  1.009  0.763    1.323
9    2005  33  0.982  0.763    1.310
10   1998  26  0.976  0.756    1.292
11   1996  24  0.981  0.767    1.278
12   1995  23  0.960  0.755    1.272
13   2009  37  0.949  0.751    1.261
14   1997  25  0.953  0.756    1.261
15   2010 38  0.870  0.728    1.194
16   2007 35  0.881  0.758    1.162
17   1994 22  0.878  0.763    1.150
18   1993 21  0.502  0.736    0.682
19   2011 39  0.118  0.720    0.163
```

In this case, 2000 still ranks as Ramirez’s best season relative to his peers, but notice that his 1999 season has fallen from 2nd to 3rd. Since by definition a league batter has an OPS+ of 1, Ramirez posted 17 consecutive seasons with an OPS that was at least 15% better than the average across the major leagues—a truly impressive feat.

Finally, not all joins are the same. An `inner_join()` requires corresponding entries in both tables. Conversely, a `left_join()` returns at least as many rows as there are in the first table, regardless of whether there are matches in the second table. Thus, an `inner_join()` is bidirectional, whereas in a `left_join()`, the order in which you specify the tables matters.

Consider the career of Cal Ripken, who played in 21 seasons from 1981 to 2001. His career overlapped with Ramirez’s in the nine seasons from 1993 to 2001, so for those, the league averages we computed before are useful.

```r
ripken <- Batting %>% filter(playerID == "ripkeca01")
nrow(inner_join(ripken, mlb, by = c("yearID" = "yearID")))

[1] 9

nrow(inner_join(mlb, ripken, by = c("yearID" = "yearID"))) # same

[1] 9
```

For seasons when Ramirez did not play, NA’s will be returned.
CHAPTER 4. DATA WRANGLING

Conversely, by reversing the order of the tables in the join, we return the 19 seasons for which we have already computed the league averages, regardless of whether there is a match for Ripken (results not displayed).

4.5 Further resources

Hadley Wickham is an enormously influential innovator in the field of statistical computing. Along with his colleagues at RStudio and other organizations, he has made significant contributions to improve data wrangling in R. These packages are sometimes called the “Hadleyverse” or the “tidyverse,” and are now manageable through a single tidyverse package. His papers and vignettes describing widely used packages such as dplyr and tidyr are highly recommended reading. In particular, his paper on tidy data builds upon notions of normal forms—common to database designers from computer science—to describe a process of thinking about how data should be stored and formatted. Finzer writes of a “data habit of mind” that needs to be inculcated among data scientists. The RStudio data wrapping cheat sheet is a useful reference.

Sean Lahman, a self-described “database journalist,” has long curated his baseball data set, which feeds the popular website baseball-reference.com. Michael Friendly maintains the Lahman R package. For the baseball enthusiast, Cleveland Indians analyst Max Marchi and Jim Albert have written an excellent book on analyzing baseball data in R. Albert has also written a book describing how baseball can be used as a motivating example for teaching statistics.

4.6 Exercises

Exercise 4.1

Each of these tasks can be performed using a single data verb. For each task, say which verb it is:

1. Find the average of one of the variables.
2. Add a new column that is the ratio between two variables.
3. Sort the cases in descending order of a variable.
4. Create a new data table that includes only those cases that meet a criterion.

5. From a data table with three categorical variables A, B, and C, and a quantitative variable X, produce a data frame that has the same cases but only the variables A and X.

**Exercise 4.2**

Use the `nycflights13` package and the `flights` data frame to answer the following questions: What month had the highest proportion of cancelled flights? What month had the lowest? Interpret any seasonal patterns.

**Exercise 4.3**

Use the `nycflights13` package and the `flights` data frame to answer the following question: What plane (specified by the `tailnum` variable) traveled the most times from New York City airports in 2013? Plot the number of trips per week over the year.

**Exercise 4.4**

Use the `nycflights13` package and the `flights` and `planes` tables to answer the following questions: What is the oldest plane (specified by the `tailnum` variable) that flew from New York City airports in 2013? How many airplanes that flew from New York City are included in the `planes` table?

**Exercise 4.5**

Use the `nycflights13` package and the `flights` and `planes` tables to answer the following questions: How many planes have a missing date of manufacture? What are the five most common manufacturers? Has the distribution of manufacturer changed over time as reflected by the airplanes flying from NYC in 2013? (Hint: you may need to recode the manufacturer name and collapse rare vendors into a category called **Other**.)

**Exercise 4.6**

Use the `nycflights13` package and the `weather` table to answer the following questions: What is the distribution of temperature in July, 2013? Identify any important outliers in terms of the `wind_speed` variable. What is the relationship between `dewp` and `humid`? What is the relationship between `precip` and `visib`?

**Exercise 4.7**

Use the `nycflights13` package and the `weather` table to answer the following questions: On how many days was there precipitation in the New York area in 2013? Were there differences in the mean visibility (`visib`) based on the day of the week and/or month of the year?

**Exercise 4.8**

Define two new variables in the `Teams` data frame from the `Lahman` package: batting average (BA) and slugging percentage (SLG). Batting average is the ratio of hits (H) to at-bats (AB), and slugging percentage is total bases divided by at-bats. To compute total bases, you get 1 for a single, 2 for a double, 3 for a triple, and 4 for a home run.

**Exercise 4.9**
Plot a time series of SLG since 1954 conditioned by lgID. Is slugging percentage typically higher in the American League (AL) or the National League (NL)? Can you think of why this might be the case?

**Exercise 4.10**

Display the top 15 teams ranked in terms of slugging percentage in MLB history. Repeat this using teams since 1969.

**Exercise 4.11**

The Angels have at times been called the California Angels (CAL), the Anaheim Angels (ANA), and the Los Angeles Angels of Anaheim (LAA). Find the 10 most successful seasons in Angels history. Have they ever won the World Series?

**Exercise 4.12**

Create a factor called `election` that divides the `yearID` into four-year blocks that correspond to U.S. presidential terms. During which term have the most home runs been hit?

**Exercise 4.13**

Name every player in baseball history who has accumulated at least 300 home runs (HR) and at least 300 stolen bases (SB).

**Exercise 4.14**

Name every pitcher in baseball history who has accumulated at least 300 wins (W) and at least 3,000 strikeouts (SO).

**Exercise 4.15**

Identify the name and year of every player who has hit at least 50 home runs in a single season. Which player had the lowest batting average in that season?

**Exercise 4.16**

The Relative Age Effect is an attempt to explain anomalies in the distribution of birth month among athletes. Briefly, the idea is that children born just after the age cut-off for participation will be as much as 11 months older than their fellow athletes, which is enough of a disparity to give them an advantage. That advantage will then be compounded over the years, resulting in notably more professional athletes born in these months. Display the distribution of birth months of baseball players who batted during the decade of the 2000s. How are they distributed over the calendar year? Does this support the notion of a relative age effect? Use the `Births78` data set from the `mosaicData` package as a reference.

**Exercise 4.17**

The `Violations` data set in the `mdsr` package contains information regarding the outcome of health inspections of restaurants in New York City. Use these data to calculate the median violation score by zip code for zip codes in Manhattan with 50 or more inspections. What pattern do you see between the number of inspections and the median score?

**Exercise 4.18**

Download data on the number of deaths by firearm from the Florida Department of Law Enforcement. Wrangle these data and use `ggplot2` to re-create Figure 6.1.