

Data Technologies and Computational Reasoning

Preface

This data science text aims to integrate statistical and computational reasoning in the context of problem solving with data. We examine questions in science, social science, business, industry, and government and work with data from within these contexts as we attempt to answer the questions. We hope that by reading this material and trying the exercises and projects, the reader will:

- Gain the essential skills needed to engage in data-analysis-based problem solving and computational reasoning;
- Express ideas clearly, precisely, and concisely through code;
- Apply statistical thinking throughout the data analysis cycle, from data acquisition and cleaning to organization and analysis to communicating results; and
- Build the confidence needed to overcome future computational challenges.

These are broadly stated goals, and in order to achieve them we have created a sequence of materials covering a wide range of computational and statistical topics for readers spanning the spectrum of experience from those who have never programmed to those who regularly write code and are looking for guidance in advancing their abilities and filling in gaps in their skill set. Overall, we provide an introduction to elementary aspects of programming, program non-trivial tasks, explore how to make code more efficient, and cover a variety of data technologies for compare and contrast several different technologies for accessing, cleaning, merging, and ‘munging’ data from a variety of sources. Simultaneously, we cover many statistical topics, including exploratory data analysis, visualization, resampling techniques such as the bootstrap, jackknife and cross-validation, and the Monte Carlo method. We also introduce a broad array of modern statistical and machine learning methods. The intent is that data scientists can use this material to learn about most of the topics they are expected to use in their daily work.

The material is divided into 4 parts. We briefly describe these parts and include descriptions of the learning goals for each of the 14 chapters.

I. Scripting and Exploratory Data Analysis

This first part in the data science series introduces the reader to the basics of *R* and helps them become fluent in manipulating and exploring data and creating informative visualizations. We assume no prior experience with a programming language. Yet, someone who routinely uses *R* may want to dip into this material. For example, he or she may want to read the summary boxes or a particular section that highlights the core features of the language and why they are so useful in data analysis. In this part, we attempt to present the paradigms of the language and avoid code recipes for handling data. Moreover, we model the EDA mindset as we read and examine data, and we present guidelines for

making good graphics that may interest any level of reader. We think this part can be used in an introductory course that focuses on elementary *R* programming for reading data into *R*, transforming data into different structures, practicing exploratory data analysis, and critiquing and composing informative data visualizations.

Introduction to *R*

In this chapter, we introduce the statistical programming language *R*. We demonstrate how to express computations in the *R* language and how to think about the computational paradigms of the language. Rather than simply provide the nuts and bolts and code templates, we aim to explain the essential features of the language and how to think about *R*’s computational model. This includes a description of how these features help statisticians with their work. One goal for the reader is to understand the Read-Evaluate-Print-Loop (REPL), including being able to write simple expressions and evaluate them; read and correct the syntax of an expression; and use boolean algebra and logical expressions. Another goal is to become familiar with a handful of functions in *R* that perform simple statistical summaries or provide information about a data object. This includes being able to call a function, passing in the appropriate arguments, and to read documentation for a function to determine how to invoke it. We also introduce simple data objects, such as vectors and data frames, and the primitive data types, and how to create subsets of them.

Reading and Exploring Data in Tables

This chapter on data tables and exploratory data analysis aims to get the reader handling data that are organized in plain-text, rectangular formats. We provide examples of how to read, clean, and validate data sets and the exploratory phase of data analysis. More specifically, we distinguish between various formats of plain text data and map them into rectangular data structures. These formats include comma-separated values, tab-delimited values, fixed width format, and key-value pairs. We also demonstrate how to: perform simple summaries and visualizations to validate the data have been properly read and the values are as expected; initiate simple corrections to the process of reading the data and modify the data after it has been read to, e.g., identify missing values, rename variables, convert data values to a format more suitable for analysis, and create new variables from those supplied; determine how to rearrange values in a table so that the resulting structure is more conducive to analysis—this may require stacking columns of variables that contain the same information, transposing rows of values into columns, or creating arrays of data tables.

Exploratory Data Analysis (EDA) features prominently in this chapter. We focus on how to conduct initial explorations of data, interpret the findings, consider new directions to take, and continue exploration.

Visualization

Visualization plays an important role in exploration and presentation and this chapter touches on both goals for making statistical graphs. We address the process of how to: select an appropriate plot to make based on the type of data being analyzed; describe the shape of a distribution and relationships between two variables; apply the method of comparison to examine distributions and relationships for subgroups of data, augment plots with useful information, and consider transformations of variables; bring the data provenance into consideration of the implication of the findings to other, possibly more general, settings. We introduce basic guidelines for choosing colors, scaling axes, and transforming variables in order that plots make the data stand out, facilitate comparisons, and are information rich. Additionally, we model the iterative process of how beginning with a simple plot we can continue to refine and augment it while following these guidelines, and we demonstrate how at times we must create a unique sort of plot in order to best convey our discoveries.

With the exception of one section, we present this material in a context that does not

depend on the programming language. In that section we introduce the graphics model in both base *R* and *ggplot2* and provide code examples that demonstrate how to create a few of the figures in the chapter with these two approaches.

Reading and Exploring Complex Data

This chapter on complex data structures provides examples of handling data that are not simple table-like structures. As in Chapter 2, we focus on the thought process behind decisions about how to read and organize data. Here, the data are not necessarily easily mapped into a table or data frame. The goals are to: identify semi-structured data that cannot be conveyed in simple delimited text files that naturally map to a data frame or table, e.g., *JSON*-formatted data and ragged arrays where records have varying number of observations; carry out the process of organizing these data into a structure suitable for analysis, including how to handle data at different levels of granularity and how to organize the data for analysis; and for data in an *R* list structure, determine its organization and how to access an element, take a subset, and apply a function to each element.

II. Programming, Resampling, and Monte Carlo

In this part, we cover the basics of programming, including how to write, test, and debug functions. We assume no experience in programming and identify the steps in writing a function and demonstrate several core computational paradigms, such as control flow and call frames. In subsequent chapters, our reader gains experience in programming through several examples that address how to model and make inferences from data, develop prediction methods, and understand properties of stochastic processes and statistical estimators. Included with these motivating examples is an introduction to concepts in Monte Carlo simulation of stochastic systems and to resampling techniques for statistical inference and prediction. We expect this material to be accessible to a reader with little statistics background as we attempt to construct understanding of statistical ideas through simulation and resampling. Furthermore, as we work through many data-analysis problems, we aim to convey computational reasoning and develop expertise in writing well-tested, modular code.

Programming Concepts

This chapter provides a basic introduction to programming, including how to compose a function and write expressions that control the flow of evaluation of code. We demonstrate how to map the description of a simple task into a function; we first write code for a specific example and then generalize it by defining a function, encapsulating the code we have written into the function, and identifying the inputs to the function and their default values (if any). We describe the notion of a call frame and how variables and their inputs are set up when a function is invoked, including argument matching and lazy evaluation. We augment the sequential evaluation of expressions with how to conditionally evaluate expressions and how to use a loop mechanism to repeat the evaluation of code. Also in this chapter, we explore simple debugging strategies, e.g., how to distinguish between different types of errors, decode syntax errors, and set up test cases to confirm the code works as expected. Additionally, we introduce style guidelines for writing code so that our code is clear, consistent, and readable.

Resampling for Inference and Prediction

In this chapter, the reader gains experience with programming in the context of problem solving with resampling methods. The computational topics include: modularity, where we

identify subtasks, develop small functions to carry out these tasks, and combine them to solve a data analysis problem; testing, where we develop test cases for debugging code and demonstrate how to trace call frames and browse the variables as our code executes to uncover the source of errors; profiling code to find where bottlenecks occur, and redesign our code to improve computational efficiency; and developing more complex function signatures that use, e.g., \dots and functions as parameters. The examples examine sampling distributions and approximate standard errors computed with the jackknife and bootstrap. A case study is worked where cross-validation is used in a nearest neighbor setting to select the number of neighbors.

Simulation and Stochastic Modeling

This chapter has the same computational goals as the chapter on resampling. The problems originate from trying to understand the behavior of a random quantity, such as the sampling distribution of a statistic calculated from data where the data follow an unusual distribution that arises from the manner in which the data are collected (e.g., censoring, truncation, or a mixture of 2 distributions). This chapter also includes a comprehensive case study that demonstrates the topics of modularity, testing, profiling, and efficiency. Specifically, the behavior of the Bihani-Middleton-Levine traffic model is explored through simulation. This case study is revisited in the advanced programming part to demonstrate object-oriented programming and how to interface to *C* code.

III. Data Technologies

Each chapter in this part aims to familiarize readers with an essential data technology used in data science. As with the earlier chapters that introduce *R* and programming, our philosophy is to bring to the fore the core concepts in the technology and provide examples that demonstrate how and when the technology can be used. The technologies included here address the topics of accessing data via Web scraping and APIs, relational databases, text processing, and shell tools. We introduce specific technologies related to these topics, including *HTML*, *XML* and *XPath*; *SQL*; regular expressions; and shell commands. We describe their core concepts without reference to *R*. However, in order to work through the examples and address the problems posed, we expect the reader to have basic expertise in *R* at the level introduced in the first two parts of this series.

Text Manipulation and Regular Expressions

In this chapter, we demonstrate the fundamental ideas behind the regular expression language, introduce the basic elements of the language, and show how they can be combined to express patterns. We aim to provide the reader with skills necessary to work with text data, including being able to: modify strings with simple string manipulation functionality to split up and paste strings together; read and understand the syntax of a regular expression and construct regular expressions to search for patterns in strings. We introduce meta characters to modify a pattern, create alternate patterns, and identify sub-patterns. In the examples, we clean text to create variables for analysis and perform more intensive text mining. Text mining tasks including being able to remove words from a corpus, reduce related words to their stem, tally word counts for documents, and construct measures of word frequency to compare documents.

Relational Databases and SQL

In this chapter, we describe the concept of a relational database (RDBMS), introduce the

structured query language (SQL) to extract data, and consider how to work with data in a RDBMS from within *R*. More specifically, the goals of this chapter are for the reader to able to: understand the advantages of using a relational database to store data, including issues related to security, redundancy, and efficiency; read schema and describe the organization of tables in a relational database; understand the syntax of a *SELECT* statement; perform simple extractions from a single table in a database; and merge and combine information within and across tables. We use *R* functions to interface with a database, and we draw parallels between comparable tasks for data frames in *R* and tables in a RDBMS. In addition to column-oriented data bases, we briefly address the topic of other types of data bases, such as *noSQL*, and when they are useful. These are covered in more detail in Chapter 12.

Shell Tools

In this chapter, we introduce the shell language and perform simply command-line tasks. More specifically, the reader is introduced to syntax of a shell expression, including how to specify arguments, pipe commands, and identify syntax errors. We provide examples of how to use simple shell commands to manage files, e.g., copy, move, delete, and examine file contents, and how to manage processes, including remotely accessing files and running programs, e.g., with terminal multiplexers such as *Screen* and *tmux*. Finally, we explore files and their contents through simple shell commands and combinations of these commands. We also discuss reproducibility and versioning systems in this chapter.

XML

In this chapter, the reader learns about the structure of an *XML* document, including understanding the document's hierarchy of elements and what it means to be well-formed. Importantly, we cover the concept of how to locate content in the document with *XPath* expressions and demonstrate how to extract content to create data structures amenable for analysis.

Web Technologies

In this chapter the reader learns how to scrape data from Web pages and tables, including using *XPath* to locate content in *HTML* and *XML* documents. We also introduce the ideas underlying *HTTP* and provide examples of how to access data via forms and *RESTful* Web APIs. We briefly cover the basics of authentication in Web services and pages.

IV. Data Science Projects and Case Studies

Many fundamental statistical concepts and methods appear throughout the other parts in this book. This particular part supplements the earlier parts with projects and a fully worked case study. We provide descriptions of a dozen open-ended projects that integrate material from multiple computational topics in a modern data problem. These projects and the examples found throughout the text differ from traditional computer science assignments and data analysis projects in that they require both a demonstration of computing aptitude and insightful data analysis. Many introduce computationally intensive statistical methodologies, such as recursive partitioning, nearest neighbor methods, and hierarchical clustering, that are relatively easy to understand from an algorithmic perspective, that are not typically taught until late in an undergraduate statistics curriculum if at all, and that can be exciting to apply to a data analysis problem. Other projects focus on simulation studies of a complex system. Typically, there is not one correct answer for a project, and they can be solved using very different statistical methods and approaches from those in-

cluded with the project description. For example, the naive Bayes method is described in one project as an approach for classifying spam, but recursive partitioning, which appears in the description of the indoor positioning project, can be used instead for the spam case study. Finally, also in this part, we provide a case study that demonstrates how a data scientist might tackle large and/or complex data in a reproducible and systematic way.

Student Work

This book contains 3 different types of assignments. The Guided Practice problems found at the end of each chapter serve to reinforce the basic concepts introduced in the chapter. These problems are taken from our weekly laboratory assignments that we designed for students to gain experience with the material before embarking on more open-ended homework and projects. We provide solutions for these problems including detailed descriptions and alternative coding approaches.

The section called Exercises that follows the Guided Practice consists of problems that we typically assign for homework. These problems tend to be more open-ended than the guided practice problems but more structured than the projects. Many of our homework assignments have been split into several contiguous exercises in this section. Lastly, as mentioned already, projects can be found in their own part of this series. These projects typically require mastery of multiple computational topics, synthesis of statistical and computational thinking, and learning about and applying a new statistical methodology.

Selecting Material for a Course

This series goes beyond the basics to teach the practice and process of programming and computational reasoning for data science, and the material can be combined in different ways for different courses. We have experience teaching all of this material in a sequence of 2 10-week courses on the common computational tasks of data science. The first course in this series covers much of the material in Parts I and II, and the second focusses on the technologies found in Part III. The projects in Part IV are used as motivating examples and assignments in both courses. We provide a list of topics and corresponding chapters for each of the courses in Table 0.1. Note that there is some overlap from one course to the next.

We have also covered much of the material here in a 15-week course called Concepts in Computing with Data. This course has no programming or statistics pre-requisites and is aimed at the sophomore/junior level. In Table 0.2, we provide a sample syllabus that includes a week-by-week list of topics and readings.

Available Materials

The Web site <http://rdatasciencecases.org> provides data, code, and supplementary material for this book. It also provides ideas and details for additional case studies. We hope others will contribute their case studies so that we can make these available to the community at large.

TABLE 0.1: Topic List for a 2-course (20 week) Sequence in Data Science

Session	Topic	Reading
A	Introduction to <i>R</i>	Chap 1
A	Reading Data	2.1-4, 4.1-6
A	Programming	Chap 5
A	Simulation & Resampling	6.1-2, 6.4, 7.1-2, 7.6-7
A	String Manipulation & Regular Expressions	Chap 8
B	Review Programming	Chap 5
B	Text Manipulation & Regular Expressions	Chap 8
B	Relational Databases	Chap 9
B	Shell Tools	Chap 10
B	Web Technologies	Chap 11

TABLE 0.2: Sample Syllabi for a 15-week Data Science Course


Week	Topic	Reading
1	Introduction to <i>R</i>	1.1-7, 1.13-15
2	Vectorized operations & EDA	1.8-9, 2.7
3	Graphics	3.1-5
4	Data Structures	1.10-11, 2.4-6, 4.2
5	Introduction to Programming	5.1-6
6	Simulation & Project	5.7, 7.1-3, 15.11
7	Review & Midterm	
8	Debugging & Style	5.8-9
9	Shell	2.1-3, 10.1-5
10	Regular Expressions & Text mining	8.1-5, 15.9
11	XML & Simple Web Scraping	11.2-3, 4.4
12	SQL	9.1-5
13	Project & related topics, e.g., resampling	6.1-2, 6.4
14	Project-related topics, e.g., nearest neighbor	6.4, 15.1
15	Review & Project help	

Typographic Conventions

In this series, we use other languages in addition to *R*, such as *SQL*. While the context should clearly indicate that a code block is in a language other than *R*, we also specify this in the margin of the page. For example, a *UNIX grep* command appears as

Shell `grep position2d JRSPdata_2010_03_10.log | grep -v ' 004 '`

In the process of writing code, we introduce errors and mistakes with the idea that these are useful for learning how best to approach and solve computational problems. For this reason, some of the code in this book is purposefully incorrect or ill-advised (i.e., it works but is not a good approach). We identify such code with a no-entry symbol in the page margin, e.g.,

 `createGrid(c(3, 5), .5)`
`Error in grid[pos] = sample(rep(c("red", "blue"), numCars)) :`
`(converted from warning) number of items to replace is not`
`a multiple of replacement length`

We note that the code we present for creating the figures differs slightly from the code we actually used to create the displayed figures. Furthermore, typically we add titles to our plots, either for interactive viewing or for including in presentations and reports. However, we have not done that in this book because the graphics are displayed in figures which include their own captions and titles. In order to avoid redundancy, we have eliminated the specification of a plot title in our code.

Acknowledgments

We first started teaching a course based on these ideas at our respective institutions in Spring 2005. Over the past dozen years, we have continued to develop and refine these materials, created new courses for different levels of students, and provided materials for our colleagues at our and other institutions to teach data science. We thank our colleagues for their feedback on these materials and this approach to teaching data analysis and data science in an integrated framework. In the process of developing our course materials and student assignments, we developed many case studies, which we spun off into a collection that appears as a separate publication, *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*. This collection includes contributions from others and we thank them for their contributions and impact on our thinking. Most importantly, we owe a tremendous debt of gratitude to the graduate students who have assisted in the teaching of our courses and the undergraduate students who have taken our courses and provided valuable feedback and enthusiasm. We especially thank, Gabe Becker, Erica Christianson, Clark Fitzgerald, Inna Gerlovina, Christine Ho, Neal Fultz, Karl Kumbier, Gang Liang, Bitao Liu, Jarrod Millman, Sandy Nathan, Kellie Ottoboni, Bradly Stadie, Nick Ulle, and Charlotte Wickham.

We also would like to acknowledge the grants that in part supported this work. Specifically, this material is based in part upon work supported by the National Science Foundation under Grant Numbers DUE-0618865, DMS-0840001, and DUE-1043634.

Contents

I Scripting and Exploratory Data Analysis	1
1 Introduction to R	3
1.1 Introduction	4
1.2 Getting Started	4
1.3 Computations and Expressions	4
1.3.1 Arithmetic Expressions and Order of Operations	5
1.3.2 Call Expressions	6
1.4 Variables and Assignment	7
1.5 Syntax and Parsing	9
1.6 Data Types	12
1.6.1 Finding Information on Vectors	13
1.7 Vectorized Operations	15
1.7.1 Aggregator Functions	17
1.7.2 Relational Operations	17
1.7.3 Boolean Algebra	18
1.7.4 Coercion	19
1.8 Data Frames	21
1.9 Subsets	24
1.9.1 Subsetting with Logical Vectors	24
1.9.2 Subsetting by Position	26
1.9.3 Subsetting by Exclusion	28
1.9.4 Subsetting by Name	29
1.9.5 Subsetting All	30
1.9.6 Assigning Values to Subsets	30
1.9.7 Subsetting Data Frames	31
1.10 Applying Functions to Variables in a Data Frame	32
1.11 Creating Vectors	34
1.11.1 Concatenating Values into a Vector	34
1.11.2 Creating Sequences of Values	35
1.11.3 Functions for Operating on Vectors	36
1.11.4 Functions for Manipulating Character Vectors	37
1.11.5 Creating Vectors from Different Types	37
1.12 Recycling Rules	38
1.13 Managing Your Workspace	38
1.13.1 Storing and Removing Variables	39
1.13.2 Searching for Variables	40
1.14 Getting Help	42
1.15 Software for Statisticians	42
1.16 Summary	44
1.17 Summary of Basic Functions for Working with Vectors and Data Frames	45
1.18 Guided Practice	47

xii

xii

Contents

1.19 Exercises	50
2 Reading and Exploring Data in Tables	51
2.1 Introduction	51
2.2 Formats for Tabular Data	53
2.2.1 Delimited Data	54
2.2.1.1 Comma-Delimited Traffic Data	54
2.2.2 Fixed Width Format	56
2.2.2.1 Fixed Width Formatted Drug Abuse Warning Network Survey	56
2.2.3 Key-Value Pairs	58
2.3 Validating and Cleaning the Data	59
2.3.1 Updating Variable Names and Formatting Time for Traffic	60
2.3.2 Data Types for DAWN Survey	61
2.3.3 Validating Text – Hillary Clinton’s Email	64
2.4 Selecting a Structure: Data Frame, Matrix and Array	68
2.4.1 Collections of Matrices – Handwritten Digits Data	68
2.4.1.1 Applying Functions to Matrices and Arrays	70
2.5 Reshaping Data Tables	71
2.5.1 Stacking Traffic Flow	72
2.5.2 Rearranging World Bank Country Statistics	75
2.6 Merging Data Tables	81
2.6.1 Merging Names and Emails	82
2.7 Exploratory Data Analysis	85
2.7.1 Exploring Traffic on California Freeways	86
2.7.2 Exploring Country Statistics	91
2.7.3 Exploring Emergency Room Visits due to Drug Abuse	94
2.7.4 EDA Summary	99
2.8 Summary	102
2.9 Functions for Reading and Exploring Data	103
2.10 Guided Practice	104
2.11 Exercises	105
3 Visualization	107
3.1 Introduction	107
3.1.1 Composing a Graph of Voter Registration Trends	108
3.2 Data Types and Plot Choice	111
3.2.1 Terminology	111
3.2.2 The Kaiser Family Data	111
3.2.3 Univariate Plots	113
3.2.4 Bivariate Plots	117
3.2.5 Conveying Relationships between 3 or More Variables	122
3.2.6 Scalability – Real Estate Data with 500,000 records	125
3.2.7 Measurements in Time	128
3.2.8 Geographic Data	129
3.3 Guidelines	131
3.3.1 Scale	135
3.3.2 Position	135
3.3.3 Shape	136
3.3.4 Aggregates	136
3.3.5 Color	136

Contents

xiii

3.3.6 Context	137
3.3.7 Over Arching Considerations	137
3.4 Iterative process	138
3.5 Rs Graphics Models	139
3.5.1 Painter’s Model in Base R	139
3.5.2 Grammar of Graphics Model in ggplot2	142
3.6 Creating Unique Plots	146
3.7 Summary	146
3.8 Exercises	146
4 Reading and Exploring Complex Data	149
4.1 Introduction	149
4.2 Lists	155
4.2.1 Subsetting Lists	157
4.2.1.1 Accessing An Element of a List	158
4.2.2 Applying Functions to Elements of a List	160
4.2.3 Exploring Rainfall on the Colorado Front Range	161
4.3 Reading Data into a Character Vector	166
4.3.1 A Study of Web Page Updates	167
4.4 Reading Data from a Web Page	172
4.4.1 World Records in the Men’s 1500 meter	174
4.5 Reading JSON Formatted Data	175
4.6 Kiva	181
4.6.1 Loan Elements	183
4.6.2 The Payments	186
4.6.3 The Borrowers	186
4.6.4 Final Structure	187
4.7 Summary	187
4.8 Functions for Handling Complex Data Formats	187
4.9 Guided Practice	188
4.10 Exercises	189
II Programming, Monte Carlo, and Resampling	191
5 Programming Concepts	193
5.1 Introduction	193
5.2 Steps in Writing Functions	196
5.2.1 Calculating BMI from Height and Weight	196
5.2.2 Generalizing Code	198
5.2.3 Commenting Code	199
5.2.4 Reporting Run Times in Seconds	199
5.2.5 Taking the Logarithm of ‘0’	201
5.3 Defining a Function	202
5.3.1 Anonymous Functions	204
5.4 Calling a Function	205
5.4.1 Argument Matching by Name and Position	205
5.4.2 The ... parameter	207
5.4.3 Lazy Evaluation	208
5.4.4 The Search Path	209
5.4.5 Partial Matching	211
5.5 Exiting a Function	211

xiv

Contents

5.6 Conditionally Invoking Code	212
5.6.1 Conditionally Modifying Inputs to log(0)	214
5.6.2 Converting Liquid Measures into Tbs. Using switch()	220
5.7 Iterative Evaluation of Code	221
5.7.1 Reading Thousands of Files of Data With a for Loop	221
5.7.2 Generating Fibonacci Numbers	223
5.7.3 Playing Penney’s Game	228
5.8 Style Guidelines	233
5.9 Beginning Notions of Debugging	237
5.10 Summary	240
5.11 Control Flow and Debugging Functions	240
5.12 Guided Practice	241
5.13 Exercises	244
6 Resampling for Inference and Prediction	245
6.1 Introduction	245
6.2 The Bootstrap and Inference	247
6.2.1 Percentile Bootstrap of Median House Price	247
6.2.2 Studentized Bootstrap for Repair Times	251
6.2.3 Jackknife Estimate of the Standard Error for Average Repair Time	261
6.3 Permutations and Testing for a Gender Effect	267
6.4 Cross-Validation and Prediction	275
6.4.1 Nearest Neighbor Prediction	280
6.4.2 Hold Out Test Set	290
6.4.3 v-fold Cross Validation	296
6.4.4 Lazy Evaluation	299
6.5 Summary	299
6.6 Guided Practice	299
6.7 Exercises	299
7 Simulation and Stochastic Modeling	301
7.1 Introduction	301
7.2 Monte Carlo Studies	303
7.2.1 Monte Carlo Study of Magnetic Resonance Scans	304
7.3 Random Number Generation	312
7.3.1 The Cumulative Distribution Function	313
7.3.2 Congruential Methods	313
7.4 Inverse Probability Sampling	315
7.4.1 Simulating the Censored Log-Normal	317
7.4.2 Simulating the Truncated Normal Distribution	318
7.5 Acceptance-Rejection Sampling	320
7.5.1 Tail of a Distribution	323
7.6 Simulation Study of a Location Estimator for a Mixture Distribution	326
7.7 Simulating the Biham-Middleton-Levine Model for Traffic	334
7.7.1 The Initial Traffic Configuration	335
7.7.1.1 Testing the Grid Creation Function	338
7.7.2 Displaying the Grid	341
7.7.3 Moving the Cars	345
7.7.4 Evaluating the Performance of the Code	350
7.8 Summary	358
7.9 Exercises	358

III Data Technologies	359
8 Text Manipulation and Regular Expressions	361
8.1 Introduction	361
8.2 Basic Concepts in Pattern Matching	364
8.2.1 Matching Literals	364
8.2.2 Modifiers and Meta Characters	365
8.2.3 Equivalent Characters	366
8.3 Using Regular Expressions in <i>R</i>	370
8.3.1 Writing Our Own Literal Matcher	374
8.4 Alternatives, Grouping, and Backreferences	376
8.4.1 Grouping Characters into Sub-patterns	376
8.4.2 Alternative Patterns	376
8.4.3 Back referencing a Sub-pattern	377
8.5 Greedy Matching	377
8.5.1 Lazy Matching	378
8.6 Examples	378
8.7 Summary	384
8.7.1 Summary	384
8.7.2 Resources	384
8.8 Summary of Pattern Matching and String Manipulation Functions	385
8.9 Text Mining & Natural Language Processing	386
8.10 Guided Practice	386
8.10.1 Answers	387
8.11 Exercises	392
9 Relational Databases and SQL	393
10 Shell Tools	395
10.1 Digital Information	396
10.1.1 Bits and Bytes	396
10.1.2 Representing Numbers in Binary	396
10.2 Files and File Systems	396
10.2.1 Plain Text Files	396
10.2.2 Binary Files	396
10.2.3 File Systems	396
10.2.4 Permissions	396
10.3 Basics of a Shell Command	396
10.3.1 Syntax	396
10.3.2 Getting Help	396
10.4 Managing and Navigating a File System	396
10.4.1 Wildcards	396
10.5 Redirection	396
10.6 Managing Processes with the Shell	396
10.6.1 Running in Batch	397
10.7 Remote Login	397
10.7.1 Screen/tmux	397
10.8 Cases	397
10.9 Advanced Topics	397
10.9.1 grep	397
10.9.2 cut	397

10.10Summary	397
10.11Guided Practice	397
10.12Exercises	397
11 XML	399
11.1 Overview of <i>XML</i>	399
11.2 Hierarchical Structure	407
11.3 <i>XML</i> Namespaces and Additional <i>XML</i> Elements	410
11.4 The Document Object Model (<i>DOM</i>)	412
11.5 Accessing Nodes in the <i>DOM</i>	414
11.6 <i>XPath</i> and the <i>XML</i> Tree	421
11.7 <i>XPath</i> Syntax	425
11.7.1 The Axis	426
11.7.2 The Node Test	429
11.7.3 The Predicate	429
11.8 Summary of Functions to Read <i>HTML</i> , <i>XML</i> , and <i>JSON</i> into <i>R</i> Data Frames and Lists	431
12 Web Technologies	433
12.1 Introduction	433
12.2 Before You Scrape!	434
12.3 Scraping Data from <i>HTML</i> Tables	435
12.3.1	435
12.3.1.1 <i>HTML</i> Forms	437
12.3.2 Specifying the Column Types	441
12.4 Scraping Links from <i>HTML</i> Pages	441
12.5 Overview of <i>HTML</i> and <i>XML</i>	441
12.6 Scraping Arbitrary Content from <i>HTML</i> Pages with <i>XPath</i>	441
12.6.1 Finding Patterns in the <i>HTML</i> Content and using <i>XPath</i>	441
12.7 Scraping Data from <i>HTML</i> Forms	441
12.8 Dynamic <i>JavaScript</i> Content with Selenium	441
12.9 Basics of <i>HTTP</i> Queries	441
12.10REST-based Web Services	441
12.10.1 <i>XML</i>	441
12.10.2 <i>JSON</i>	441
12.11Summary	441
12.12Guided Practice	441
12.13Exercises	441
IV Data Science Projects and Case Studies	443
13 Projects	445
13.1 Digit Recognition	445
13.1.1 Goal	445
13.1.2 Background	445
13.1.3 The Data	446
13.1.4 Tasks	447
13.1.5 k-Nearest Neighbors	448
13.1.6 Cross Validation and Model Selection	449
13.2 Adhoc Networks (Simulation Study)	451

13.3 Graphs: perhaps voting records of senators, citation networks, Enron Email (graphs)	451
13.4 Web Cache (Poisson arrival process)	451
13.5 Presidential Election (Maps)	451
13.6 NASA Satellite	451
13.7 Indoor Positioning System (Nearest neighbor and Cross-Validation)	451
13.8 Spam Filtering (classification trees)	451
13.9 State of the Union Speeches (Term Frequencies and Hierarchical Clustering and Multi-dimensional Scaling)	451
13.10 (Large Data)	451
14 Case: Reading Data with a Diverse Set of Approaches	453
Index	455