Quantitative Social Science

An Introduction

KOSUKE IMAI

PRINCETON UNIVERSITY PRESS
Princeton and Oxford

Epigraph on page 189 from Tukey, John W., 1977, *Exploratory data analysis.* Reading, Mass: Addison-Wesley Publishing, an imprint of Pearson Education, Inc.

Published by Princeton University Press, 41 William Street,
Princeton, New Jersey 08540

In the United Kingdom: Princeton University Press, 6 Oxford Street,
Woodstock, Oxfordshire OX20 1TR

press.princeton.edu

ISBN 978-0-691-16703-9
ISBN (pbk.) 978-0-691-17546-1

Library of Congress Control Number 2016962298

British Library Cataloging-in-Publication Data is available

This book has been composed in Minion Pro and Univers for display
Printed on acid-free paper ∞

Typeset by Nova Techset Pvt Ltd, Bangalore, India
Printed in the United States of America

10 9 8 7 6 5 4 3

To Christina, Keiji, and Misaki

Contents

List of Tables

List of Figures

Preface

I decided to write this book in order to convince the next generation of students and researchers that data analysis is a powerful tool for answering many important and interesting questions about societies and human behavior. Today's societies confront a number of challenging problems, including those in economics, politics, education, and public health. Data-driven approaches are useful for solving these problems, and we need more talented individuals to work in this area. I hope that this book will entice young students and researchers into the fast-growing field of quantitative social science.

This book grew out of the two undergraduate courses I have taught at Princeton over the last several years: POL 245: Visualizing Data and POL 345: Quantitative Analysis and Politics. While teaching these courses, I realized that students need to be exposed to exciting ideas from actual quantitative social science research as early in the course as possible. For this reason, unlike traditional introductory statistics textbooks, this book features data analysis from the very beginning, using examples directly taken from published social science research. The book provides readers with extensive data analysis experience before introducing probability and statistical theories. The idea is that by the time they reach those challenging chapters, readers will understand why those materials are necessary in order to conduct quantitative social science research.

The book starts with a discussion of causality in both experimental and observational studies using the examples of racial discrimination and get-out-the-vote campaigns. We then cover measurement and prediction as two other primary goals of data analysis in social science research. The book also includes a chapter on the analysis of textual, network, and spatial data, giving readers a glimpse of modern quantitative social science research. Probability and statistical theories are introduced after these data analysis chapters. The mathematical level of the book is kept to a minimum, and neither calculus nor linear algebra is used. However, the book introduces probability and statistical theories in a conceptually rigorous manner so that readers can understand the underlying logic.

This book would not exist without support from a number of individuals. I would like to thank my colleagues at Princeton, especially those in the Dean of the College's office and the McGraw Center for Teaching and Learning, for their generous support. I was one of the first beneficiaries of the 250th Anniversary Fund for Teaching Innovation in Undergraduate Education. I thank Liz Colagiuri, Khristina Gonzalez, Lisa Herschbach, Clayton Marsh, Diane McKay, and Nic Voge, who trusted my ambitious vision of how introductory data analysis and statistics should be taught.

They allowed me to design a course at the Freshman Scholars Institute (FSI), and many of the ideas in this book were born there. The FSI is a great diversity initiative for first-generation college students, and I am proud to be a part of it. I am also grateful to Princeton University administrators for their generous support for my teaching initiatives. They include Jill Dolan, Chris Eisgruber, Dave Lee, Nolan McCarty, Debbie Prentice, and Val Smith.

I especially thank my coinstructors who helped me develop the materials included in this book. James Lo, Jonathan Olmsted, and Will Lowe made significant contributions to POL 245 taught at FSI. I was fortunate to have an amazing group of graduate students who served as teaching assistants for my courses. They include Alex Acs, Jaquilyn Waddell Boie, Will Bullock, Munji Choi, Winston Chou, Elisha Cohen, Brandon de la Cuesta, Ted Enamorado, Matt Incantalupo, Tolya Levshin, Asya Magazinnik, Carlos Velasco Rivera, Alex Tarr, Bella Wang, and Teppei Yamamoto, several of whom won teaching awards for their incredible work. Evan Chow and Hubert Jin contributed to the creation of **swirl** exercises. Other students, including Alessia Azermadhi, Naoki Egami, Tyler Pratt, and Arisa Wada, helped me develop materials at various stages of this book project.

During the production phase of this book, the following individuals gave me detailed comments and suggestions that have significantly improved the presentation: Jaquilyn Waddell Boie, Lauren Konken, Katie McCabe, Grace Rehaut, Ruby Shao, and Tyler Simko. Without their contributions, this book would have looked quite different. I also thank at least several hundred students at Princeton and many other institutions who used an earlier version of this book. Their extensive feedback has helped me revise the manuscript. I also thank Neal Beck, Andy Hall, Ryan Moore, and Marc Ratkovic for their comments on earlier versions of the manuscript. I wish to thank Eric Crahan and Brigitte Pelner of Princeton University Press for guiding me through the publication process.

Several people had a significant impact on how this book is written. My graduate school adviser, Gary King, taught me everything, from how to conduct quantitative social science research to how to teach statistics to social scientists. Although more than a decade has passed since I left Harvard, Gary has always been someone to whom I can turn for advice and support. Three of my Princeton colleagues—Christina Davis, Amaney Jamal, and Evan Liebenman—formed the team "old dogs learning new tricks" and took the three-course graduate quantitative methods sequence. Their willingness to patiently sit through my lectures gave me new motivation. They also set a great example for young researchers that even senior scholars should continue learning. Interactions with them during those classes gave me new insights about how statistical methods should be taught.

My deepest gratitude goes to my family. My mother, Fumiko, my father, Takashi, and my brother, Mineki, have always encouraged me to pursue my dreams regardless of what they are. Although we now live on opposite sides of the globe, every day I feel lucky to have such a wonderful family. My parents-in-law, Al and Carole Davis, have been supportive of me since the mid-1990s when I first came to the United States without being able to speak or understand much English. They have always made me feel at home and part of their family. My two wonderful children, Keiji and Misaki, have been a source of joy and happiness. However difficult my work is, their beautiful

smiles remind me what the most important things are in my life. Finally, I dedicate this book to my wife, Christina, who has been the best partner and a constant source of inspiration for more than two decades. Christina encouraged me to write this book, and as always I am glad to have followed her advice. Even though one never observes counterfactuals, I can say with confidence that I have lived and will continue to live life to the fullest because of our partnership.

Kosuke Imai
November 2016
Princeton, New Jersey

Quantitative Social Science

Chapter 1

Introduction

In God we trust; all others must bring data.
—William Edwards Deming

Quantitative social science is an interdisciplinary field encompassing a large number of disciplines, including economics, education, political science, public policy, psychology, and sociology. In quantitative social science research, scholars analyze data to understand and solve problems about society and human behavior. For example, researchers examine racial discrimination in the labor market, evaluate the impact of new curricula on students' educational achievements, predict election outcomes, and analyze social media usage. Similar data-driven approaches have been taken up in other neighboring fields such as health, law, journalism, linguistics, and even literature. Because social scientists directly investigate a wide range of real-world issues, the results of their research have enormous potential to directly influence individual members of society, government policies, and business practices.

Over the last couple of decades, quantitative social science has flourished in a variety of areas at an astonishing speed. The number of academic journal articles that present empirical evidence from data analysis has soared. Outside academia, many organizations—including corporations, political campaigns, news media, and government agencies—increasingly rely on data analysis in their decision-making processes. Two transformative technological changes have driven this rapid growth of quantitative social science. First, the Internet has greatly facilitated the *data revolution*, leading to a spike in the amount and diversity of available data. Information sharing makes it possible for researchers and organizations to disseminate numerous data sets in digital form. Second, the *computational revolution*, in terms of both software and hardware, means that anyone can conduct data analysis using their personal computer and favorite data analysis software.

As a direct consequence of these technological changes, the sheer volume of data available to quantitative social scientists has rapidly grown. In the past, researchers largely relied upon data published by governmental agencies (e.g., censuses, election outcomes, and economic indicators) as well as a small number of data sets collected by research groups (e.g., survey data from national election studies and hand-coded data sets about war occurrence and democratic institutions). These data sets still

play an important role in empirical analysis. However, the wide variety of new data has significantly expanded the horizon of quantitative social science research. Researchers are designing and conducting randomized experiments and surveys on their own. Under pressure to increase transparency and accountability, government agencies are making more data publicly available online. For example, in the United States, anyone can download detailed data on campaign contributions and lobbying activities to their personal computers. In Nordic countries like Sweden, a wide range of registers, including income, tax, education, health, and workplace, are available for academic research.

New data sets have emerged across diverse areas. Detailed data about consumer transactions are available through electronic purchasing records. International trade data are collected at the product level between many pairs of countries over several decades. Militaries have also contributed to the data revolution. During the Afghanistan war in the 2000s, the United States and international forces gathered data on the geo-location, timing, and types of insurgent attacks and conducted data analysis to guide counterinsurgency strategy. Similarly, governmental agencies and nongovernmental organizations collected data on civilian casualties from the war. Political campaigns use data analysis to devise voter mobilization strategies by targeting certain types of voters with carefully selected messages.

These data sets also come in varying forms. Quantitative social scientists are analyzing digitized texts as data, including legislative bills, newspaper articles, and the speeches of politicians. The availability of social media data through websites, blogs, tweets, SMS messaging, and Facebook has enabled social scientists to explore how people interact with one another in the online sphere. Geographical information system (GIS) data sets are also widespread. They enable researchers to analyze the legislative redistricting process or civil conflict with attention paid to spatial location. Others have used satellite imagery data to measure the level of electrification in rural areas of developing countries. While still rare, images, sounds, and even videos can be analyzed using quantitative methods for answering social science questions.

Together with the revolution of information technology, the availability of such abundant and diverse data means that anyone, from academics to practitioners, from business analysts to policy makers, and from students to faculty, can make data-driven discoveries. In the past, only statisticians and other specialized professionals conducted data analysis. Now, everyone can turn on their personal computer, download data from the Internet, and analyze them using their favorite software. This has led to increased demands for accountability to demonstrate policy effectiveness. In order to secure funding and increase legitimacy, for example, nongovernmental organizations and governmental agencies must now demonstrate the efficacy of their policies and programs through rigorous evaluation.

This shift towards greater transparency and data-driven discovery requires that students in the social sciences learn how to analyze data, interpret the results, and effectively communicate their empirical findings. Traditionally, introductory statistics courses have focused on teaching students basic statistical concepts by having them conduct straightforward calculations with paper and pencil or, at best, a scientific calculator. Although these concepts are still important and covered in this book, this

traditional approach cannot meet the current demands of society. It is simply not sufficient to achieve "statistical literacy" by learning about common statistical concepts and methods. Instead, all students in the social sciences should acquire basic data analysis skills so that they can exploit the ample opportunities to learn from data and make contributions to society through data-driven discovery.

The belief that everyone should be able to analyze data is the main motivation for writing this book. The book introduces the three elements of data analysis required for quantitative social science research: research contexts, programming techniques, and statistical methods. Any of these elements in isolation is insufficient. Without research contexts, we cannot assess the credibility of assumptions required for data analysis and will not be able to understand what the empirical findings imply. Without programming techniques, we will not be able to analyze data and answer research questions. Without the guidance of statistical principles, we cannot distinguish systematic patterns, known as signals, from idiosyncratic ones, known as noise, possibly leading to invalid inference. (Here, inference refers to drawing conclusions about unknown quantities based on observed data.) This book demonstrates the power of data analysis by combining these three elements.

1.1 Overview of the Book

This book is written for anyone who wishes to learn data analysis and statistics for the first time. The target audience includes researchers, undergraduate and graduate students in social science and other fields, as well as practitioners and even ambitious high-school students. The book has no prerequisite other than some elementary algebra. In particular, readers do not have to possess knowledge of calculus or probability. No programming experience is necessary, though it can certainly be helpful. The book is also appropriate for those who have taken a traditional "paper-and-pencil" introductory statistics course where little data analysis is taught. Through this book, students will discover the excitement that data analysis brings. Those who want to learn R programming might also find this book useful, although here the emphasis is on how to use R to answer quantitative social science questions.

As mentioned above, the unique feature of this book is the presentation of programming techniques and statistical concepts simultaneously through analysis of data sets taken directly from published quantitative social science research. The goal is to demonstrate how social scientists use data analysis to answer important questions about societal problems and human behavior. At the same time, users of the book will learn fundamental statistical concepts and basic programming skills. Most importantly, readers will gain experience with data analysis by examining approximately forty data sets.

The book consists of eight chapters. The current introductory chapter explains how to best utilize the book and presents a brief introduction to R, a popular open-source statistical programming environment. R is freely available for download and runs on Macintosh, Windows, and Linux computers. Readers are strongly encouraged to use RStudio, another freely available software package that has numerous features to make data analysis easier. This chapter ends with two exercises that are

designed to help readers practice elementary R functionalities using data sets from published social science research. All data sets used in this book are freely available for download via links from `http://press.princeton.edu/qss/`. Links to other useful materials, such as the review exercises for each chapter, can also be found on the website. With the exception of chapter 5, the book focuses on the most basic syntax of R and does not introduce the wide range of additional packages that are available. However, upon completion of this book, readers will have acquired enough R programming skills to be able to utilize these packages.

Chapter 2 introduces *causality*, which plays an essential role in social science research whenever we wish to find out whether a particular policy or program changes an outcome of interest. Causality is notoriously difficult to study because we must infer counterfactual outcomes that are not observable. For example, in order to understand the existence of racial discrimination in the labor market, we need to know whether an African-American candidate who did not receive a job offer would have done so if they were white. We will analyze the data from a well-known experimental study in which researchers sent the résumés of fictitious job applicants to potential employers after randomly choosing applicants' names to sound either African-American or Caucasian. Using this study as an application, the chapter will explain how the randomization of treatment assignment enables researchers to identify the average causal effect of the treatment.

Additionally, readers will learn about causal inference in observational studies where researchers do not have control over treatment assignment. The main application is a classic study whose goal was to figure out the impact of increasing the minimum wage on employment. Many economists argue that a minimum-wage increase can reduce employment because employers must pay higher wages to their workers and are therefore made to hire fewer workers. Unfortunately, the decision to increase the minimum wage is not random, but instead is subject to many factors, like economic growth, that are themselves associated with employment. Since these factors influence which companies find themselves in the treatment group, a simple comparison between those who received treatment and those who did not can lead to biased inference.

We introduce several strategies that attempt to reduce this type of selection bias in observational studies. Despite the risk that we will inaccurately estimate treatment effects in observational studies, the results of such studies are often easier to generalize than those obtained from randomized controlled trials. Other examples in chapter 2 include a field experiment concerning social pressure in get-out-the-vote mobilization. Exercises then include a randomized experiment that investigates the causal effect of small class size in early education as well as a natural experiment about political leader assassination and its effects. In terms of R programming, chapter 2 covers logical statements and subsetting.

Chapter 3 introduces the fundamental concept of *measurement*. Accurate measurement is important for any data-driven discovery because bias in measurement can lead to incorrect conclusions and misguided decisions. We begin by considering how to measure public opinion through sample surveys. We analyze the data from a study in which researchers attempted to measure the degree of support among Afghan citizens for international forces and the Taliban insurgency during the Afghanistan war. The chapter explains the power of randomization in survey sampling. Specifically,

random sampling of respondents from a population allows us to obtain a representative sample. As a result, we can infer the opinion of an entire population by analyzing one small representative group. We also discuss the potential biases of survey sampling. Nonresponses can compromise the representativeness of a sample. Misreporting poses a serious threat to inference, especially when respondents are asked sensitive questions, such as whether they support the Taliban insurgency.

The second half of chapter 3 focuses on the measurement of latent or unobservable concepts that play a key role in quantitative social science. Prominent examples of such concepts include ability and ideology. In the chapter, we study political ideology. We first describe a model frequently used to infer the ideological positions of legislators from roll call votes, and examine how the US Congress has polarized over time. We then introduce a basic clustering algorithm, k-means, that makes it possible for us to find groups of similar observations. Applying this algorithm to the data, we find that in recent years, the ideological division within Congress has been mainly characterized by the party line. In contrast, we find some divisions within each party in earlier years. This chapter also introduces various measures of the spread of data, including quantiles, standard deviation, and the Gini coefficient. In terms of R programming, the chapter introduces various ways to visualize univariate and bivariate data. The exercises include the reanalysis of a controversial same-sex marriage experiment, which raises issues of academic integrity while illustrating methods covered in the chapter.

Chapter 4 considers *prediction*. Predicting the occurrence of certain events is an essential component of policy and decision-making processes. For example, the forecasting of economic performance is critical for fiscal planning, and early warnings of civil unrest allow foreign policy makers to act proactively. The main application of this chapter is the prediction of US presidential elections using preelection polls. We show that we can make a remarkably accurate prediction by combining multiple polls in a straightforward manner. In addition, we analyze the data from a psychological experiment in which subjects are shown the facial pictures of unknown political candidates and asked to rate their competence. The analysis yields the surprising result that a quick facial impression can predict election outcomes. Through this example, we introduce linear regression models, which are useful tools to predict the values of one variable based on another variable. We describe the relationship between linear regression and correlation, and examine the phenomenon called "regression towards the mean," which is the origin of the term "regression."

Chapter 4 also discusses when regression models can be used to estimate causal effects rather than simply make predictions. Causal inference differs from standard prediction in requiring the prediction of counterfactual, rather than observed, outcomes using the treatment variable as the predictor. We analyze the data from a randomized natural experiment in India where randomly selected villages reserved some of the seats in their village councils for women. Exploiting this randomization, we investigate whether or not having female politicians affects policy outcomes, especially concerning the policy issues female voters care about. The chapter also introduces the regression discontinuity design for making causal inference in observational studies. We investigate how much of British politicians' accumulated wealth is due to holding political office. We answer this question by comparing those who barely won an election with those who narrowly lost it. The chapter introduces powerful but

challenging R programming concepts: loops and conditional statements. The exercises at the end of the chapter include an analysis of whether betting markets can precisely forecast election outcomes.

Chapter 5 is about the *discovery* of patterns from data of various types. When analyzing "big data," we need automated methods and visualization tools to identify consistent patterns in the data. First, we analyze texts as data. Our primary application here is authorship prediction of *The Federalist Papers*, which formed the basis of the US Constitution. Some of the papers have known authors while others do not. We show that by analyzing the frequencies of certain words in the papers with known authorship, we can predict whether Alexander Hamilton or James Madison authored each of the papers with unknown authorship. Second, we show how to analyze network data, focusing on explaining the relationships among units. Within marriage networks in Renaissance Florence, we quantify the key role played by the Medici family. As a more contemporary example, various measures of centrality are introduced and applied to social media data generated by US senators on Twitter.

Finally in chapter 5, we introduce geo-spatial data. We begin by discussing the classic spatial data analysis conducted by John Snow to examine the cause of the 1854 cholera outbreak in London. We then demonstrate how to visualize spatial data through the creation of maps, using US election data as an example. For spatial–temporal data, we create a series of maps as an animation in order to visually characterize changes in spatial patterns over time. Thus, the chapter applies various data visualization techniques using several specialized R packages.

Chapter 6 shifts the focus from data analysis to *probability*, a unified mathematical model of uncertainty. While earlier chapters examine how to estimate parameters and make predictions, they do not discuss the level of uncertainty in empirical findings, a topic that chapter 7 introduces. Probability is important because it lays a foundation for statistical inference, the goal of which is to quantify inferential uncertainty. We begin by discussing the question of how to interpret probability from two dominant perspectives, frequentist and Bayesian. We then provide mathematical definitions of probability and conditional probability, and introduce several fundamental rules of probability. One such rule is called Bayes' rule. We show how to use Bayes' rule and accurately predict individual ethnicity using surname and residence location when no survey data are available.

This chapter also introduces the important concepts of random variables and probability distributions. We use these tools to add a measure of uncertainty to election predictions that we produced in chapter 4 using preelection polls. Another exercise adds uncertainty to the forecasts of election outcomes based on betting market data. The chapter concludes by introducing two fundamental theorems of probability: the law of large numbers and the central limit theorem. These two theorems are widely applicable and help characterize how our estimates behave over repeated sampling as sample size increases. The final set of exercises then addresses two problems: the German cryptography machine from World War II (Enigma), and the detection of election fraud in Russia.

Chapter 7 discusses how to quantify the *uncertainty* of our estimates and predictions. In earlier chapters, we introduced various data analysis methods to find

patterns in data. Building on the groundwork laid in chapter 6, chapter 7 thoroughly explains how certain we should be about such patterns. This chapter shows how to distinguish signals from noise through the computation of standard errors and confidence intervals as well as the use of hypothesis testing. In other words, the chapter concerns statistical inference. Our examples come from earlier chapters, and we focus on measuring the uncertainty of these previously computed estimates. They include the analysis of preelection polls, randomized experiments concerning the effects of class size in early education on students' performance, and an observational study assessing the effects of a minimum-wage increase on employment. When discussing statistical hypothesis tests, we also draw attention to the dangers of multiple testing and publication bias. Finally, we discuss how to quantify the level of uncertainty about the estimates derived from a linear regression model. To do this, we revisit the randomized natural experiment of female politicians in India and the regression discontinuity design for estimating the amount of wealth British politicians are able to accumulate by holding political office.

The final chapter concludes by briefly describing the next steps readers might take upon completion of this book. The chapter also discusses the role of data analysis in quantitative social science research.

1.2 How to Use this Book

In this section, we explain how to use this book, which is based on the following principle:

> One can learn data analysis only by doing, not by reading.

This book is not just for reading. The emphasis must be placed on gaining experience in analyzing data. This is best accomplished by trying out the code in the book on one's own, playing with it, and working on various exercises that appear at the end of each chapter. All code and data sets used in the book are freely available for download via links from `http://press.princeton.edu/qss/`.

The book is cumulative. Later chapters assume that readers are already familiar with most of the materials covered in earlier parts. Hence, in general, it is not advisable to skip chapters. The exception is chapter 5, "Discovery," the contents of which are not used in subsequent chapters. Nevertheless, this chapter contains some of the most interesting data analysis examples of the book and readers are encouraged to study it.

The book can be used for course instruction in a variety of ways. In a traditional introductory statistics course, one can assign the book, or parts of it, as supplementary reading that provides data analysis exercises. The book is best utilized in a data analysis course where an instructor spends less time on lecturing to students and instead works interactively with students on data analysis exercises in the classroom. In such a course, the relevant portion of the book is assigned prior to each class. In the classroom, the instructor reviews new methodological and programming concepts and then applies them to one of the exercises from the book or any other similar application of their choice. Throughout this process, the instructor can discuss the exercises interactively

with students, perhaps using the Socratic method, until the class collectively arrives at a solution. After such a classroom discussion, it would be ideal to follow up with a computer lab session, in which a small number of students, together with an instructor, work on another exercise.

This teaching format is consistent with the "particular general particular" principle.[1] This principle states that an instructor should first introduce a particular example to illustrate a new concept, then provide a general treatment of it, and finally apply it to another particular example. Reading assignments introduce a particular example and a general discussion of new concepts to students. Classroom discussion then allows the instructor to provide another general treatment of these concepts and then, together with students, apply them to another example. This is an effective teaching strategy that engages students with active learning and builds their ability to conduct data analysis in social science research. Finally, the instructor can assign another application as a problem set to assess whether students have mastered the materials. To facilitate this, for each chapter instructors can obtain, upon request, access to a private repository that contains additional exercises and their solutions.

In terms of the materials to cover, an example of the course outline for a 15-week-long semester is given below. We assume that there are approximately two hours of lectures and one hour of computer lab sessions each week. Having hands-on computer lab sessions with a small number of students, in which they learn how to analyze data, is essential.

Chapter title	*Chapter number*	*Weeks*
Introduction	1	1
Causality	2	2–3
Measurement	3	4–5
Prediction	4	6–7
Discovery	5	8–9
Probability	6	10–12
Uncertainty	7	13–15

For a shorter course, there are at least two ways to reduce the material. One option is to focus on aspects of data science and omit statistical inference. Specifically, from the above outline, we can remove chapter 6, "Probability," and chapter 7, "Uncertainty." An alternative approach is to skip chapter 5, "Discovery," which covers the analysis of textual, network, and spatial data, and include the chapters on probability and uncertainty.

Finally, to ensure mastery of the basic methodological and programming concepts introduced in each chapter, we recommend that users first read a chapter, practice all of the code it contains, and upon completion of each chapter, try the online review questions before attempting to solve the associated exercises. These review questions

[1] Frederick Mosteller (1980) "Classroom and platform performance." *American Statistician*, vol. 34, no. 1 (February), pp. 11–17.

Table 1.1. The **swirl** Review Exercises.

Chapter	***swirl** lesson*	*Sections covered*
1: Introduction	INTRO1	1.3
	INTRO2	1.3
2: Causality	CAUSALITY1	2.1–2.4
	CAUSALITY2	2.5–2.6
3: Measurement	MEASUREMENT1	3.1–3.4
	MEASUREMENT2	3.5–3.7
4: Prediction	PREDICTION1	4.1
	PREDICTION2	4.2
	PREDICTION3	4.3
5: Discovery	DISCOVERY1	5.1
	DISCOVERY2	5.2
	DISCOVERY3	5.3
6: Probability	PROBABILITY1	6.1–6.3
	PROBABILITY2	6.4–6.5
7: Uncertainty	UNCERTAINTY1	7.1
	UNCERTAINTY2	7.2
	UNCERTAINTY3	7.3

Note: The table shows the correspondence between the chapters and sections of the book and each set of **swirl** review exercises.

are available as **swirl** lessons via links from `http://press.princeton.edu/qss/`, and can be answered within R. Instructors are strongly encouraged to assign these **swirl** exercises prior to each class so that students learn the basics before moving on to more complicated data analysis exercises. To start the online review questions, users must first install the **swirl** package (see section 1.3.7) and then the lessons for this book using the following three lines of commands within R. Note that this installation needs to be done only once.

```
install.packages("swirl") # install the package
library(swirl) # load the package
install_course_github("kosukeimai", "qss-swirl") # install the course
```

Table 1.1 lists the available set of **swirl** review exercises along with their corresponding chapters and sections. To start a **swirl** lesson for review questions, we can use the following command.

```
library(swirl)
swirl()
```

More information about **swirl** is available at `http://swirlstats.com/`.

1.3 Introduction to R

This section provides a brief, self-contained introduction to R that is a prerequisite for the remainder of this book. R is an open-source statistical programming environment, which means that anyone can download it for free, examine source code, and make their own contributions. R is powerful and flexible, enabling us to handle a variety of data sets and create appealing graphics. For this reason, it is widely used in academia and industry. The *New York Times* described R as

> a popular programming language used by a growing number of data analysts inside corporations and academia. It is becoming their lingua franca... whether being used to set ad prices, find new drugs more quickly or fine-tune financial models. Companies as diverse as Google, Pfizer, Merck, Bank of America, the InterContinental Hotels Group and Shell use it.... "The great beauty of R is that you can modify it to do all sorts of things," said Hal Varian, chief economist at Google. "And you have a lot of prepackaged stuff that's already available, so you're standing on the shoulders of giants."[2]

To obtain R, visit `https://cran.r-project.org/` (The Comprehensive R Archive Network or CRAN), select the link that matches your operating system, and then follow the installation instructions.

While a powerful tool for data analysis, R's main cost from a practical viewpoint is that it must be learned as a programming language. This means that we must master various syntaxes and basic rules of computer programming. Learning computer programming is like becoming proficient in a foreign language. It requires a lot of practice and patience, and the learning process may be frustrating. Through numerous data analysis exercises, this book will teach you the basics of statistical programming, which then will allow you to conduct data analysis on your own. The core principle of the book is that we can learn data analysis only by analyzing data.

Unless you have prior programming experience (or have a preference for another text editor such as Emacs), we recommend that you use RStudio. RStudio is an open-source and free program that greatly facilitates the use of R. In one window, RStudio gives users a text editor to write programs, a graph viewer that displays the graphics we create, the R console where programs are executed, a help section, and many other features. It may look complicated at first, but RStudio can make learning how to use R much easier. To obtain RStudio, visit `http://www.rstudio.com/` and follow the download and installation instructions. Figure 1.1 shows a screenshot of RStudio.

In the remainder of this section, we cover three topics: (1) using R as a calculator, (2) creating and manipulating various objects in R, and (3) loading data sets into R.

1.3.1 ARITHMETIC OPERATIONS

We begin by using R as a calculator with standard arithmetic operators. In figure 1.1, the left-hand window of RStudio shows the R console where we can directly enter R

[2] Vance, Ashlee. 2009. "Data Analysts Captivated by R's Power." *New York Times*, January 6.

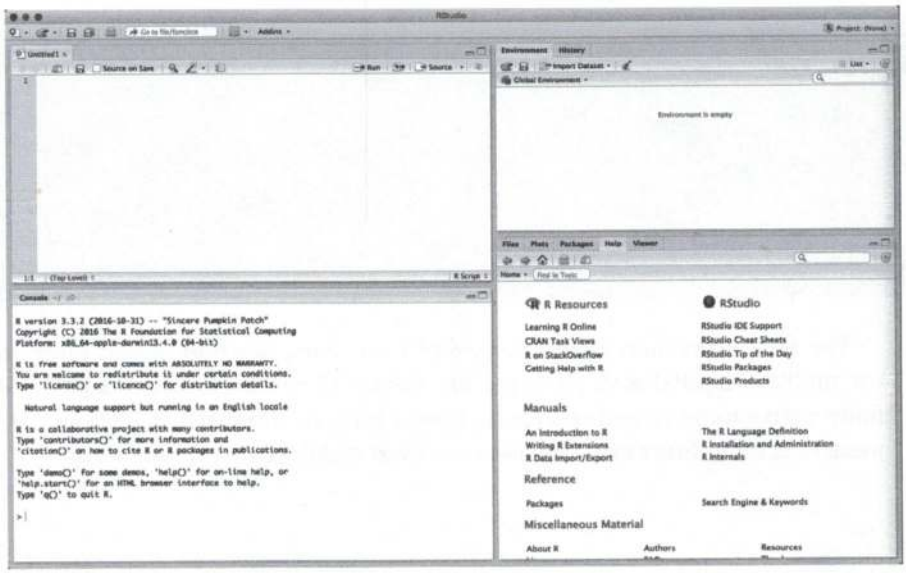


Figure 1.1. Screenshot of RStudio (version 1.0.44). The upper-left window displays a script that contains code. The lower-left window shows the console where R commands can be directly entered. The upper-right window lists R objects and a history of executed R commands. Finally, the lower-right window enables us to view plots, data sets, files and subdirectories in the working directory, R packages, and help pages.

commands. In this R console, we can type in, for example, `5 + 3`, then hit Enter on our keyboard.

```
5 + 3
## [1] 8
```

R ignores spaces, and so `5+3` will return the same result. However, we added a space before and after the operator + to make it easier to read. As this example illustrates, this book displays R commands followed by the outputs they would produce if entered in the R console. These outputs begin with ## to distinguish them from the R commands that produced them, though this mark will not appear in the R console. Finally, in this example, `[1]` indicates that the output is the first element of a *vector* of length 1 (we will discuss vectors in section 1.3.3). It is important for readers to try these examples on their own. Remember that we can learn programming only by doing! Let's try other examples.

```
5 - 3
## [1] 2
5 / 3
## [1] 1.666667
5 ^ 3
```

```
## [1] 125
5 * (10 - 3)
## [1] 35
sqrt(4)
## [1] 2
```

The final expression is an example of a so-called *function*, which takes an input (or multiple inputs) and produces an output. Here, the function `sqrt()` takes a nonnegative number and returns its square root. As discussed in section 1.3.4, R has numerous other functions, and users can even make their own functions.

1.3.2 OBJECTS

R can store information as an *object* with a name of our choice. Once we have created an object, we just refer to it by name. That is, we are using objects as "shortcuts" to some piece of information or data. For this reason, it is important to use an intuitive and informative name. The name of our object must follow certain restrictions. For example, it cannot begin with a number (but it can contain numbers). Object names also should not contain spaces. We must avoid special characters such as `%` and `$`, which have specific meanings in R. In RStudio, in the upper-right window, called `Environment` (see figure 1.1), we will see the objects we created. We use the *assignment operator* `<-` to assign some value to an object.

For example, we can store the result of the above calculation as an object named `result`, and thereafter we can access the value by referring to the object's name. By default, R will print the value of the object to the console if we just enter the object name and hit Enter. Alternatively, we can explicitly print it by using the `print()` function.

```
result <- 5 + 3
result
## [1] 8
print(result)
## [1] 8
```

Note that if we assign a different value to the same object name, then the value of the object will be changed. As a result, we must be careful not to overwrite previously assigned information that we plan to use later.

```
result <- 5 - 3
result
## [1] 2
```

Another thing to be careful about is that object names are case sensitive. For example, `Hello` is not the same as either `hello` or `HELLO`. As a consequence, we receive an error in the R console when we type `Result` rather than `result`, which is defined above.

```
Result
## Error in eval(expr, envir, enclos): object 'Result' not found
```

Encountering programming errors or bugs is part of the learning process. The tricky part is figuring out how to fix them. Here, the error message tells us that the `Result` object does not exist. We can see the list of existing objects in the `Environment` tab in the upper-right window (see figure 1.1), where we will find that the correct object is `result`. It is also possible to obtain the same list by using the `ls()` function.

So far, we have assigned only numbers to an object. But R can represent various other types of values as objects. For example, we can store a string of characters by using quotation marks.

```
kosuke <- "instructor"
kosuke
## [1] "instructor"
```

In character strings, spacing is allowed.

```
kosuke <- "instructor and author"
kosuke
## [1] "instructor and author"
```

Notice that R treats numbers like characters when we tell it to do so.

```
Result <- "5"
Result
## [1] "5"
```

However, arithmetic operations like addition and subtraction cannot be used for character strings. For example, attempting to divide or take a square root of a character string will result in an error.

```
Result / 3
## Error in Result/3: non-numeric argument to binary operator
```

```
sqrt(Result)
## Error in sqrt(Result): non-numeric argument to mathematical function
```

R recognizes different types of objects by assigning each object to a *class*. Separating objects into classes allows R to perform appropriate operations depending on the objects' class. For example, a number is stored as a numeric object whereas a character string is recognized as a character object. In RStudio, the `Environment` window will show the class of an object as well as its name. The function (which by the way is another class) `class()` tells us to which class an object belongs.

```
result
## [1] 2
class(result)
## [1] "numeric"
Result
## [1] "5"
class(Result)
## [1] "character"

class(sqrt)
## [1] "function"
```

There are many other classes in R, some of which will be introduced throughout this book. In fact, it is even possible to create our own object classes.

1.3.3 VECTORS

We present the simplest (but most inefficient) way of entering data into R. Table 1.2 contains estimates of world population (in thousands) over the past several decades. We can enter these data into R as a numeric vector object. A *vector* or a one-dimensional array simply represents a collection of information stored in a specific order. We use the function `c()`, which stands for "concatenate," to enter a data vector containing multiple values with commas separating different elements of the vector we are creating. For example, we can enter the world population estimates as elements of a single vector.

```
world.pop <- c(2525779, 3026003, 3691173, 4449049, 5320817, 6127700,
   6916183)
world.pop
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

Table 1.2. World Population Estimates.

Year	*World population* (thousands)
1950	2,525,779
1960	3,026,003
1970	3,691,173
1980	4,449,049
1990	5,320,817
2000	6,127,700
2010	6,916,183

Source: United Nations, Department of Economic and Social Affairs, Population Division (2013). *World Population Prospects: The 2012 Revision, DVD Edition.*

We also note that the `c()` function can be used to combine multiple vectors.

```
pop.first <- c(2525779, 3026003, 3691173)
pop.second <- c(4449049, 5320817, 6127700, 6916183)
pop.all <- c(pop.first, pop.second)
pop.all
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

To access specific elements of a vector, we use square brackets `[ ]`. This is called *indexing*. Multiple elements can be extracted via a vector of indices within square brackets. Also within square brackets the dash, `-`, removes the corresponding element from a vector. Note that none of these operations change the original vector.

```
world.pop[2]
## [1] 3026003

world.pop[c(2, 4)]
## [1] 3026003 4449049

world.pop[c(4, 2)]
## [1] 4449049 3026003

world.pop[-3]
## [1] 2525779 3026003 4449049 5320817 6127700 6916183
```

Since each element of this vector is a numeric value, we can apply arithmetic operations to it. The operations will be repeated for each element of the vector. Let's

give the population estimates in millions instead of thousands by dividing each element of the vector by 1000.

```
pop.million <- world.pop / 1000
pop.million
## [1] 2525.779 3026.003 3691.173 4449.049 5320.817 6127.700
## [7] 6916.183
```

We can also express each population estimate as a proportion of the 1950 population estimate. Recall that the 1950 estimate is the first element of the vector `world.pop`.

```
pop.rate <- world.pop / world.pop[1]
pop.rate
## [1] 1.000000 1.198047 1.461400 1.761456 2.106604 2.426063
## [7] 2.738238
```

In addition, arithmetic operations can be done using multiple vectors. For example, we can calculate the percentage increase in population for each decade, defined as the increase over the decade divided by its beginning population. For example, suppose that the population was 100 thousand in one year and increased to 120 thousand in the following year. In this case, we say, "the population increased by 20%." To compute the percentage increase for each decade, we first create two vectors, one without the first decade and the other without the last decade. We then subtract the second vector from the first vector. Each element of the resulting vector equals the population increase. For example, the first element is the difference between the 1960 population estimate and the 1950 estimate.

```
pop.increase <- world.pop[-1] - world.pop[-7]
percent.increase <- (pop.increase / world.pop[-7]) * 100
percent.increase
## [1] 19.80474 21.98180 20.53212 19.59448 15.16464 12.86752
```

Finally, we can also replace the values associated with particular indices by using the usual assignment operator (<-). Below, we replace the first two elements of the `percent.increase` vector with their rounded values.

```
percent.increase[c(1, 2)] <- c(20, 22)
percent.increase
## [1] 20.00000 22.00000 20.53212 19.59448 15.16464 12.86752
```

1.3.4 FUNCTIONS

Functions are important objects in R and perform a wide range of tasks. A function often takes multiple input objects and returns an output object. We have already seen

several functions: `sqrt()`, `print()`, `class()`, and `c()`. In R, a function generally runs as `funcname(input)` where `funcname` is the function name and `input` is the input object. In programming (and in math), we call these inputs *arguments*. For example, in the syntax `sqrt(4)`, `sqrt` is the function name and 4 is the argument or the input object.

Some basic functions useful for summarizing data include `length()` for the length of a vector or equivalently the number of elements it has, `min()` for the *minimum* value, `max()` for the *maximum* value, `range()` for the *range* of data, `mean()` for the *mean*, and `sum()` for the *sum* of the data. Right now we are inputting only one object into these functions so we will not use argument names.

```
length(world.pop)
## [1] 7

min(world.pop)
## [1] 2525779

max(world.pop)
## [1] 6916183

range(world.pop)
## [1] 2525779 6916183

mean(world.pop)
## [1] 4579529

sum(world.pop) / length(world.pop)
## [1] 4579529
```

The last expression gives another way of calculating the mean as the sum of all the elements divided by the number of elements.

When multiple arguments are given, the syntax looks like `funcname(input1, input2)`. The order of inputs matters. That is, `funcname(input1, input2)` is different from `funcname(input2, input1)`. To avoid confusion and problems stemming from the order in which we list arguments, it is also a good idea to specify the name of the argument that each input corresponds to. This looks like `funcname(arg1 = input1, arg2 = input2)`.

For example, the `seq()` function can generate a vector composed of an increasing or decreasing sequence. The first argument `from` specifies the number to start from; the second argument `to` specifies the number at which to end the sequence; the last argument `by` indicates the interval to increase or decrease by. We can create an object for the `year` variable from table 1.2 using this function.

```
year <- seq(from = 1950, to = 2010, by = 10)
year
## [1] 1950 1960 1970 1980 1990 2000 2010
```

Notice how we can switch the order of the arguments without changing the output because we have named the input objects.

```
seq(to = 2010, by = 10, from = 1950)
## [1] 1950 1960 1970 1980 1990 2000 2010
```

Although not relevant in this particular example, we can also create a decreasing sequence using the `seq()` function. In addition, the colon operator `:` creates a simple sequence, beginning with the first number specified and increasing or decreasing by 1 to the last number specified.

```
seq(from = 2010, to = 1950, by = -10)
## [1] 2010 2000 1990 1980 1970 1960 1950

2008:2012
## [1] 2008 2009 2010 2011 2012

2012:2008
## [1] 2012 2011 2010 2009 2008
```

The `names()` function can access and assign names to elements of a vector. Element names are not part of the data themselves, but are helpful attributes of the R object. Below, we see that the object `world.pop` does not yet have the names attribute, with `names(world.pop)` returning the `NULL` value. However, once we assign the `year` as the labels for the object, each element of `world.pop` is printed with an informative label.

```
names(world.pop)
## NULL

names(world.pop) <- year
names(world.pop)
## [1] "1950" "1960" "1970" "1980" "1990" "2000" "2010"
```

```
world.pop
##    1950    1960    1970    1980    1990    2000    2010
## 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

In many situations, we want to create our own functions and use them repeatedly. This allows us to avoid duplicating identical (or nearly identical) sets of code chunks, making our code more efficient and easily interpretable. The `function()` function can create a new function. The syntax takes the following form.

```
myfunction <- function(input1, input2, ..., inputN) {

    DEFINE "output" USING INPUTS

    return(output)
}
```

In this example code, `myfunction` is the function name, `input1, input2, ..., inputN` are the input arguments, and the commands within the braces `{ }` define the actual function. Finally, the `return()` function returns the output of the function. We begin with a simple example, creating a function to compute a summary of a numeric vector.

```
my.summary <- function(x){ # function takes one input
  s.out <- sum(x)
  l.out <- length(x)
  m.out <- s.out / l.out
  out <- c(s.out, l.out, m.out) # define the output
  names(out) <- c("sum", "length", "mean") # add labels
  return(out) # end function by calling output
}
z <- 1:10
my.summary(z)
##     sum length    mean
##    55.0   10.0     5.5

my.summary(world.pop)
##       sum    length      mean
## 32056704         7   4579529
```

Note that objects (e.g., `x`, `s.out`, `l.out`, `m.out`, and `out` in the above example) can be defined within a function independently of the environment in which the function is being created. This means that we need not worry about using identical names for objects inside a function and those outside it.

1.3.5 DATA FILES

So far, the only data we have used has been manually entered into R. But, most of the time, we will load data from an external file. In this book, we will use the following two data file types:

- *CSV* or comma-separated values files represent tabular data. This is conceptually similar to a spreadsheet of data values like those generated by Microsoft Excel or Google Spreadsheet. Each observation is separated by line breaks and each field within the observation is separated by a comma, a tab, or some other character or string.
- *RData* files represent a collection of R objects including data sets. These can contain multiple R objects of different kinds. They are useful for saving intermediate results from our R code as well as data files.

Before interacting with data files, we must ensure they reside in the *working directory*, which R will by default load data from and save data to. There are different ways to change the working directory. In RStudio, the default working directory is shown in the bottom-right window under the `Files` tab (see figure 1.1). Oftentimes, however, the default directory is not the directory we want to use. To change the working directory, click on `More > Set As Working Directory` after choosing the folder we want to work from. Alternatively, we can use the RStudio pull-down menu `Session > Set Working Directory > Choose Directory...` and pick the folder we want to work from. Then, we will see our files and folders in the bottom-right window.

It is also possible to change the working directory using the `setwd()` function by specifying the full path to the folder of our choice as a character string. To display the current working directory, use the function `getwd()` without providing an input. For example, the following syntax sets the working directory to `qss/INTRO` and confirms the result (we suppress the output here).

```
setwd("qss/INTRO")
getwd()
```

Suppose that the United Nations population data in table 1.2 are saved as a CSV file `UNpop.csv`, which resembles that below:

```
year, world.pop
1950, 2525779
1960, 3026003
1970, 3691173
1980, 4449049
1990, 5320817
2000, 6127700
2010, 6916183
```

In RStudio, we can read in or load CSV files by going to the drop-down menu in the upper-right window (see figure 1.1) and clicking `Import Dataset > From Text`

File ... Alternatively, we can use the `read.csv()` function. The following syntax loads the data as a data frame object (more on this object below).

```
UNpop <- read.csv("UNpop.csv")
class(UNpop)
## [1] "data.frame"
```

On the other hand, if the same data set is saved as an object in an RData file named `UNpop.RData`, then we can use the `load()` function, which will load all the R objects saved in `UNpop.RData` into our R session. We do not need to use the assignment operator with the `load()` function when reading in an RData file because the R objects stored in the file already have object names.

```
load("UNpop.RData")
```

Note that R can access any file on our computer if the full location is specified. For example, we can use syntax such as `read.csv("Documents/qss/INTRO/UNpop.csv")` if the data file `UNpop.csv` is stored in the directory `Documents/qss/INTRO/`. However, setting the working directory as shown above allows us to avoid tedious typing.

A data frame object is a collection of vectors, but we can think of it like a spreadsheet. It is often useful to visually inspect data. We can view a spreadsheet-like representation of data frame objects in RStudio by double-clicking on the object name in the `Environment` tab in the upper-right window (see figure 1.1). This will open a new tab displaying the data. Alternatively, we can use the `View()` function, which as its main argument takes the name of a data frame to be examined. Useful functions for this object include `names()` to return a vector of variable names, `nrow()` to return the number of rows, `ncol()` to return the number of columns, `dim()` to combine the outputs of `ncol()` and `nrow()` into a vector, and `summary()` to produce a summary.

```
names(UNpop)
## [1] "year"       "world.pop"

nrow(UNpop)
## [1] 7

ncol(UNpop)
## [1] 2

dim(UNpop)
## [1] 7 2
```

```
summary(UNpop)
##       year        world.pop
##  Min.   :1950   Min.   :2525779
##  1st Qu.:1965   1st Qu.:3358588
##  Median :1980   Median :4449049
##  Mean   :1980   Mean   :4579529
##  3rd Qu.:1995   3rd Qu.:5724258
##  Max.   :2010   Max.   :6916183
```

Notice that the `summary()` function yields, for each variable in the data frame object, the minimum value, the first *quartile* (or 25th *percentile*), the *median* (or 50th percentile), the third quartile (or 75th percentile), and the maximum value. See section 2.6 for more discussion.

The `$` operator is one way to access an individual variable from within a data frame object. It returns a vector containing the specified variable.

```
UNpop$world.pop
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

Another way of retrieving individual variables is to use indexing inside square brackets `[ ]`, as done for a vector. Since a data frame object is a two-dimensional array, we need two indexes, one for rows and the other for columns. Using brackets with a comma `[rows, columns]` allows users to call specific rows and columns by either row/column numbers or row/column names. If we use row/column numbers, sequencing functions covered above, i.e., `:` and `c()`, will be useful. If we do not specify a row (column) index, then the syntax will return all rows (columns). Below are some examples, demonstrating the syntax of indexing.

```
UNpop[, "world.pop"] # extract the column called "world.pop"
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183

UNpop[c(1, 2, 3),]   # extract the first three rows (and all columns)
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173

UNpop[1:3, "year"]   # extract the first three rows of the "year" column
## [1] 1950 1960 1970
```

When extracting specific observations from a variable in a data frame object, we provide only one index since the variable is a vector.

```
## take elements 1, 3, 5, ... of the "world.pop" variable
UNpop$world.pop[seq(from = 1, to = nrow(UNpop), by = 2)]
## [1] 2525779 3691173 5320817 6916183
```

In R, missing values are represented by NA. When applied to an object with missing values, functions may or may not automatically remove those values before performing operations. We will discuss the details of handling missing values in section 3.2. Here, we note that for many functions, like mean(), the argument na.rm = TRUE will remove missing data before operations occur. In the example below, the eighth element of the vector is missing, and one cannot calculate the mean until R has been instructed to remove the missing data.

```
world.pop <- c(UNpop$world.pop, NA)
world.pop
## [1] 2525779 3026003 3691173 4449049 5320817 6127700  6916183
## [8]      NA

mean(world.pop)
## [1] NA

mean(world.pop, na.rm = TRUE)
## [1] 4579529
```

1.3.6 SAVING OBJECTS

The objects we create in an R session will be temporarily saved in the *workspace*, which is the current working environment. As mentioned earlier, the ls() function displays the names of all objects currently stored in the workspace. In RStudio, all objects in the workspace appear in the Environment tab in the upper-right corner. However, these objects will be lost once we terminate the current session. This can be avoided if we save the workspace at the end of each session as an RData file.

When we quit R, we will be asked whether we would like to save the workspace. We should answer no to this so that we get into the habit of explicitly saving only what we need. If we answer yes, then R will save the entire workspace as .RData in the working directory without an explicit file name and automatically load it next time we launch R. This is not recommended practice, because the .RData file is invisible to users of many operating systems and R will not tell us what objects are loaded unless we explicitly issue the ls() function.

In RStudio, we can save the workspace by clicking the Save icon in the upper-right Environment window (see figure 1.1). Alternatively, from the navigation bar, click

on `Session > Save Workspace As...`, and then pick a location to save the file. Be sure to use the file extension `.RData`. To load the same workspace the next time we start RStudio, click the `Open File` icon in the upper-right `Environment` window, select `Session > Load Workspace...`, or use the `load()` function as before.

It is also possible to save the workspace using the `save.image()` function. The file extension `.RData` should always be used at the end of the file name. Unless the full path is specified, objects will be saved to the working directory. For example, the following syntax saves the workspace as `Chapter1.RData` in the `qss/INTRO` directory provided that this directory already exists.

```
save.image("qss/INTRO/Chapter1.RData")
```

Sometimes, we wish to save only a specific object (e.g., a data frame object) rather than the entire workspace. This can be done with the `save()` function as in `save(xxx, file = "yyy.RData")`, where `xxx` is the object name and `yyy.RData` is the file name. Multiple objects can be listed, and they will be stored as a single RData file. Here are some examples of syntax, in which we again assume the existence of the `qss/INTRO` directory.

```
save(UNpop, file = "Chapter1.RData")
save(world.pop, year, file = "qss/INTRO/Chapter1.RData")
```

In other cases, we may want to save a data frame object as a CSV file rather than an RData file. We can use the `write.csv()` function by specifying the object name and the file name, as the following example illustrates.

```
write.csv(UNpop, file = "UNpop.csv")
```

Finally, to access objects saved in the RData file, simply use the `load()` function as before.

```
load("Chapter1.RData")
```

1.3.7 PACKAGES

One of R's strengths is the existence of a large community of R users who contribute various functionalities as R packages. These packages are available through the Comprehensive R Archive Network (CRAN; `http://cran.r-project.org`). Throughout the book, we will employ various packages. For the purpose of illustration, suppose that we wish to load a data file produced by another statistical software package such as Stata or SPSS. The **foreign** package is useful when dealing with files from other statistical software.

To use the package, we must load it into the workspace using the `library()` function. In some cases, a package needs to be installed before being loaded. In RStudio, we can do this by clicking on `Packages > Install` in the bottom-right window (see figure 1.1), where all currently installed packages are listed, after choosing the desired packages to be installed. Alternatively, we can install from the R console using the `install.packages()` function (the output is suppressed below). Package installation needs only to occur once, though we can update the package later upon the release of a new version (by clicking `Update` or reinstalling it via the `install.packages()` function).

```
install.packages("foreign") # install package
library("foreign") # load package
```

Once the package is loaded, we can use the appropriate functions to load the data file. For example, the `read.dta()` and `read.spss()` functions can read Stata and SPSS data files, respectively (the following syntax assumes the existence of the `UNpop.dta` and `UNpop.sav` files in the working directory).

```
read.dta("UNpop.dta")
read.spss("UNpop.sav")
```

As before, it is also possible to save a data frame object as a data file that can be directly loaded into another statistical software package. For example, the `write.dta()` function will save a data frame object as a Stata data file.

```
write.dta(UNpop, file = "UNpop.dta")
```

1.3.8 PROGRAMMING AND LEARNING TIPS

We conclude this brief introduction to R by providing several practical tips for learning how to program in the R language. First, we should use a text editor like the one that comes with RStudio to write our program rather than directly typing it into the R console. If we just want to see what a command does, or quickly calculate some quantity, we can go ahead and enter it directly into the R console. However, for more involved programming, it is always better to use the text editor and save our code as a text file with the `.R` file extension. This way, we can keep a record of our program and run it again whenever necessary.

In RStudio, use the pull-down menu `File > New File > R Script` or click the `New File` icon (a white square with a green circle enclosing a white plus sign) and choose `R Script`. Either approach will open a blank document for text editing in the upper-left window where we can start writing our code (see figure 1.2). To run our code from the RStudio text editor, simply highlight the code and press the `Run` icon. Alternatively, in Windows, Ctrl+Enter works as a shortcut. The equivalent shortcut for Mac is Command+Enter. Finally, we can also run the entire code in the background

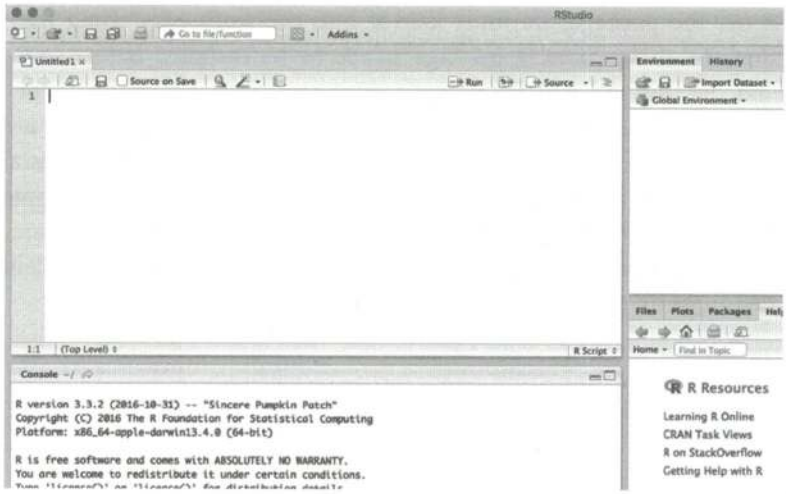


Figure 1.2. Screenshot of the RStudio Text Editor. Once we open an R script file in RStudio, the text editor will appear as one of the windows. It can then be used to write our code.

(so, the code will not appear in the console) by clicking the `Source` icon or using the `source()` function with the code file name (including a full path if it is not placed in the working directory) as the input.

```
source("UNpop.R")
```

Second, we can annotate our R code so that it can be easily understandable to ourselves and others. This is especially important as our code gets more complex. To do this, we use the comment character #, which tells R to ignore everything that follows it. It is customary to use a double comment character ## if a comment occupies an entire line and use a single comment character # if a comment is made within a line after an R command. An example is given here.

```
##
## File: UNpop.R
## Author: Kosuke Imai
## The code loads the UN population data and saves it as a Stata file
##

library(foreign)
UNpop <- read.csv("UNpop.csv")
UNpop$world.pop <- UNpop$world.pop / 1000  # population in millions
write.dta(UNpop, file = "UNpop.dta")
```

Third, for further clarity it is important to follow a certain set of coding rules. For example, we should use informative names for files, variables, and functions. Systematic spacing and indentation are essential too. In the above examples, we place spaces around all binary operators such as <-, =, +, and -, and always add a space after a comma. While comprehensive coverage of coding style is beyond the scope of this book, we encourage you to follow a useful R style guide published by Google at `https://google.github.io/styleguide/Rguide.xml`. In addition, it is possible to check our R code for potential errors and incorrect syntax. In computer science, this process is called *linting*. The `lint()` function in the **lintr** package enables the linting of R code. The following syntax implements the linting of the `UNpop.R` file shown above, where we replace the assignment operator <- in line 8 with the equality sign = for the sake of illustration.

```
library(lintr)
lint("UNpop.R")

## UNpop.R:7:7: style: Use <-, not =, for assignment.
## UNpop = read.csv("UNpop.csv")
##       ^
```

Finally, R Markdown via the **rmarkdown** package is useful for quickly writing documents using R. R Markdown enables us to easily embed R code and its output within a document using straightforward syntax in a plain-text format. The resulting documents can be produced in the form of HTML, PDF, or even Microsoft Word. Because R Markdown embeds R code as well as its output, the results of data analysis presented in documents are reproducible. R Markdown is also integrated into RStudio, making it possible to produce documents with a single click. For a quick start, see `http://rmarkdown.rstudio.com/`.

1.4 Summary

This chapter began with a discussion of the important role that quantitative social science research can play in today's data-rich society. To make contributions to this society through data-driven discovery, we must learn how to analyze data, interpret the results, and communicate our findings to others. To start our journey, we presented a brief introduction to R, which is a powerful programming language for data analysis. The remaining pages of this chapter are dedicated to exercises, designed to ensure that you have mastered the contents of this section. Start with the **swirl** review questions that are available via links from `http://press.princeton.edu/qss/`. If you answer these questions incorrectly, be sure to go back to the relevant sections and review the materials before moving on to the exercises.

Table 1.3. US Election Turnout Data.

Variable	*Description*
`year`	election year
`ANES`	ANES estimated turnout rate
`VEP`	voting eligible population (in thousands)
`VAP`	voting age population (in thousands)
`total`	total ballots cast for highest office (in thousands)
`felons`	total ineligible felons (in thousands)
`noncitizens`	total noncitizens (in thousands)
`overseas`	total eligible overseas voters (in thousands)
`osvoters`	total ballots counted by overseas voters (in thousands)

1.5 Exercises

1.5.1 BIAS IN SELF-REPORTED TURNOUT

Surveys are frequently used to measure political behavior such as voter turnout, but some researchers are concerned about the accuracy of self-reports. In particular, they worry about possible *social desirability bias* where, in postelection surveys, respondents who did not vote in an election lie about not having voted because they may feel that they should have voted. Is such a bias present in the American National Election Studies (ANES)? ANES is a nationwide survey that has been conducted for every election since 1948. ANES is based on face-to-face interviews with a nationally representative sample of adults. Table 1.3 displays the names and descriptions of variables in the `turnout.csv` data file.

1. Load the data into R and check the dimensions of the data. Also, obtain a summary of the data. How many observations are there? What is the range of years covered in this data set?

2. Calculate the turnout rate based on the voting age population or VAP. Note that for this data set, we must add the total number of eligible overseas voters since the `VAP` variable does not include these individuals in the count. Next, calculate the turnout rate using the voting eligible population or VEP. What difference do you observe?

3. Compute the differences between the VAP and ANES estimates of turnout rate. How big is the difference on average? What is the range of the differences? Conduct the same comparison for the VEP and ANES estimates of voter turnout. Briefly comment on the results.

4. Compare the VEP turnout rate with the ANES turnout rate separately for presidential elections and midterm elections. Note that the data set excludes the year 2006. Does the bias of the ANES estimates vary across election types?

Table 1.4. Fertility and Mortality Estimate Data.

Variable	*Description*
`country`	abbreviated country name
`period`	period during which data are collected
`age`	age group
`births`	number of births (in thousands), i.e., the number of children born to women of the age group
`deaths`	number of deaths (in thousands)
`py.men`	person-years for men (in thousands)
`py.women`	person-years for women (in thousands)

Source: United Nations, Department of Economic and Social Affairs, Population Division (2013). *World Population Prospects: The 2012 Revision, DVD Edition.*

5. Divide the data into half by election years such that you subset the data into two periods. Calculate the difference between the VEP turnout rate and the ANES turnout rate separately for each year within each period. Has the bias of ANES increased over time?

6. ANES does not interview prisoners and overseas voters. Calculate an adjustment to the 2008 VAP turnout rate. Begin by subtracting the total number of ineligible felons and noncitizens from the VAP to calculate an adjusted VAP. Next, calculate an adjusted VAP turnout rate, taking care to subtract the number of overseas ballots counted from the total ballots in 2008. Compare the adjusted VAP turnout with the unadjusted VAP, VEP, and the ANES turnout rate. Briefly discuss the results.

1.5.2 UNDERSTANDING WORLD POPULATION DYNAMICS

Understanding population dynamics is important for many areas of social science. We will calculate some basic demographic quantities of births and deaths for the world's population from two time periods: 1950 to 1955 and 2005 to 2010. We will analyze the following CSV data files: `Kenya.csv`, `Sweden.csv`, and `World.csv`. The files contain population data for Kenya, Sweden, and the world, respectively. Table 1.4 presents the names and descriptions of the variables in each data set. The data are collected for a period of 5 years where *person-year* is a measure of the time contribution of each person during the period. For example, a person who lives through the entire 5-year period contributes 5 person-years, whereas someone who lives only through the first half of the period contributes 2.5 person-years. Before you begin this exercise, it would be a good idea to directly inspect each data set. In R, this can be done with the `View()` function, which takes as its argument the name of the data frame to be examined. Alternatively, in RStudio, double-clicking a data frame in the `Environment` tab will enable you to view the data in a spreadsheet-like form.

1. We begin by computing *crude birth rate* (CBR) for a given period. The CBR is defined as

$$\text{CBR} = \frac{\text{number of births}}{\text{number of person-years lived}}.$$

Compute the CBR for each period, separately for Kenya, Sweden, and the world. Start by computing the total person-years, recorded as a new variable within each existing data frame via the $ operator, by summing the person-years for men and women. Then, store the results as a vector of length 2 (CBRs for two periods) for each region with appropriate labels. You may wish to create your own function for the purpose of efficient programming. Briefly describe patterns you observe in the resulting CBRs.

2. The CBR is easy to understand but contains both men and women of all ages in the denominator. We next calculate the *total fertility rate* (TFR). Unlike the CBR, the TFR adjusts for age compositions in the female population. To do this, we need to first calculate the *age-specific fertility rate* (ASFR), which represents the fertility rate for women of the reproductive age range [15, 50). The ASFR for the age range $[x, x+\delta)$, where x is the starting age and δ is the width of the age range (measured in years), is defined as

$$\text{ASFR}_{[x,\, x+\delta)} = \frac{\text{number of births to women of age } [x,\ x+\delta)}{\text{number of person-years lived by women of age } [x,\ x+\delta)}.$$

Note that square brackets, [and], include the limit whereas parentheses, (and), exclude it. For example, [20, 25) represents the age range that is greater than or equal to 20 years old and less than 25 years old. In typical demographic data, the age range δ is set to 5 years. Compute the ASFR for Sweden and Kenya as well as the entire world for each of the two periods. Store the resulting ASFRs separately for each region. What does the pattern of these ASFRs say about reproduction among women in Sweden and Kenya?

3. Using the ASFR, we can define the TFR as the average number of children that women give birth to if they live through their entire reproductive age:

$$\text{TFR} = \text{ASFR}_{[15,\, 20)} \times 5 + \text{ASFR}_{[20,\, 25)} \times 5 + \cdots + \text{ASFR}_{[45,\, 50)} \times 5.$$

We multiply each age-specific fertility rate by 5 because the age range is 5 years. Compute the TFR for Sweden and Kenya as well as the entire world for each of the two periods. As in the previous question, continue to assume that the reproductive age range of women is [15, 50). Store the resulting two TFRs for each country or the world as vectors of length 2. In general, how has the number of women changed in the world from 1950 to 2000? What about the total number of births in the world?

4. Next, we will examine another important demographic process: death. Compute the *crude death rate* (CDR), which is a concept analogous to the CBR, for each

period and separately for each region. Store the resulting CDRs for each country and the world as vectors of length 2. The CDR is defined as

$$\text{CDR} = \frac{\text{number of deaths}}{\text{number of person-years lived}}.$$

Briefly describe the patterns you observe in the resulting CDRs.

5. One puzzling finding from the previous question is that the CDR for Kenya during the period 2005–2010 is about the same level as that for Sweden. We would expect people in developed countries like Sweden to have a lower death rate than those in developing countries like Kenya. While it is simple and easy to understand, the CDR does not take into account the age composition of a population. We therefore compute the *age-specific death rate* (ASDR). The ASDR for age range $[x, x + \delta)$ is defined as

$$\text{ASDR}_{[x,\, x+\delta)} = \frac{\text{number of deaths for people of age } [x,\ x+\delta)}{\text{number of person-years of people of age } [x,\ x+\delta)}.$$

Calculate the ASDR for each age group, separately for Kenya and Sweden, during the period 2005–2010. Briefly describe the pattern you observe.

6. One way to understand the difference in the CDR between Kenya and Sweden is to compute the counterfactual CDR for Kenya using Sweden's population distribution (or vice versa). This can be done by applying the following alternative formula for the CDR:

$$\text{CDR} = \text{ASDR}_{[0,5)} \times P_{[0,5)} + \text{ASDR}_{[5,10)} \times P_{[5,10)} + \cdots,$$

where $P_{[x,x+\delta)}$ is the proportion of the population in the age range $[x, x + \delta)$. We compute this as the ratio of person-years in that age range relative to the total person-years across all age ranges. To conduct this counterfactual analysis, we use $\text{ASDR}_{[x,x+\delta)}$ from Kenya and $P_{[x,x+\delta)}$ from Sweden during the period 2005–2010. That is, first calculate the age-specific population proportions for Sweden and then use them to compute the counterfactual CDR for Kenya. How does this counterfactual CDR compare with the original CDR of Kenya? Briefly interpret the result.

Chapter 2

Causality

Shallow men believe in luck, believe in circumstances.
Strong men believe in cause and effect.
—Ralph Waldo Emerson, *The Conduct of Life*

In this chapter, we consider causality, one of the most central concepts of quantitative social science. Much of social science research is concerned with the causal effects of various policies and other societal factors. Do small class sizes raise students' standardized test scores? Would universal health care improve the health and finances of the poor? What makes voters turn out in elections and determines their choice of candidates? To answer these causal questions, one must infer a counterfactual outcome and compare it with what actually happens (i.e., a factual outcome). We show how careful research design and data analysis can shed light on these causal questions that shape important academic and policy debates. We begin with a study of racial discrimination in the labor market. We then introduce various research designs useful for causal inference and apply them to additional studies concerning social pressure and voter turnout, as well as the impact of minimum-wage increases on employment. We also learn how to subset data in different ways and compute basic descriptive statistics in R.

2.1 Racial Discrimination in the Labor Market

Does racial discrimination exist in the labor market? Or, should racial disparities in the unemployment rate be attributed to other factors such as racial gaps in educational attainment? To answer this question, two social scientists conducted the following experiment.[1] In response to newspaper ads, the researchers sent out résumés of fictitious job candidates to potential employers. They varied only the names of job applicants, while leaving the other information in the résumés unchanged.

[1] This section is based on Marianne Bertrand and Sendhil Mullainathan (2004) "Are Emily and Greg more employable than Lakisha and Jamal? A field experiment on labor market discrimination." *American Economic Review*, vol. 94, no. 4, pp. 991–1013.

Table 2.1. Résumé Experiment Data.

Variable	*Description*
`firstname`	first name of the fictitious job applicant
`sex`	sex of applicant (`female` or `male`)
`race`	race of applicant (`black` or `white`)
`call`	whether a callback was made (`1` = yes, `0` = no)

For some candidates, stereotypically African-American-sounding names such as Lakisha Washington or Jamal Jones were used, whereas other résumés contained stereotypically white-sounding names, such as Emily Walsh or Greg Baker. The researchers then compared the callback rates between these two groups and examined whether applicants with stereotypically black names received fewer callbacks than those with stereotypically white names. The positions to which the applications were sent were either in sales, administrative support, clerical, or customer services.

Let's examine the data from this experiment in detail. We begin by loading the CSV data file, `resume.csv`, into R as a data frame object called `resume` using the function `read.csv()`. Table 2.1 presents the names and descriptions of the variables in this data set.

```
resume <- read.csv("resume.csv")
```

Instead of using `read.csv()`, you can also import the data set using the pull-down menu `Tools > Import Dataset > From Text File...` in RStudio.

This data frame object `resume` is an example of *experimental data*. Experimental data are collected from an experimental research design, in which a *treatment variable*, or a causal variable of interest, is manipulated in order to examine its causal effects on an *outcome variable*. In this application, the treatment refers to the race of a fictitious applicant, implied by the name given on the résumé. The outcome variable is whether the applicant receives a callback. We are interested in examining whether or not the résumés with different names yield varying callback rates.

Experimental research examines how a treatment causally affects an outcome by assigning varying values of the treatment variable to different observations, and measuring their corresponding values of the outcome variable.

```
dim(resume)
## [1] 4870    4
```

Using the `dim()` function, we can see that `resume` consists of 4870 observations and 4 variables. Each observation represents a fictitious job applicant. The outcome variable is whether the fictitious applicant received a callback from a prospective employer. The treatment variable is the race and gender of each applicant, though

more precisely the researchers were manipulating how potential employers perceive the gender and race of applicants, rather than directly manipulating those attributes.

Once imported, the data set is displayed in a spreadsheet-like format in an RStudio window. Alternatively, we can look at the first several observations of the data set using the `head()` function.

```
head(resume)
##   firstname    sex  race call
## 1   Allison female white    0
## 2   Kristen female white    0
## 3   Lakisha female black    0
## 4   Latonya female black    0
## 5    Carrie female white    0
## 6       Jay   male white    0
```

For example, the second observation contains a résumé for Kristen, identified as a white female who did not receive a callback. In addition, we can also create a summary of the data frame via the `summary()` function.

```
summary(resume)
##    firstname        sex          race
##  Tamika : 256   female:3746   black:2435
##  Anne   : 242   male  :1124   white:2435
##  Allison: 232
##  Latonya: 230
##  Emily  : 227
##  Latoya : 226
##  (Other):3457
##       call
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.08049
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
```

The summary indicates the number of résumés for each name, gender, and race as well as the overall proportion of résumés that received a callback. For example, there were 230 résumés whose applicants had the first name of "Latonya." The summary also shows that the data set contains the same number of black and white names, while there are more female than male résumés.

We can now begin to answer whether or not the résumés with African-American-sounding names are less likely to receive callbacks. To do this, we first create a

contingency table (also called a *cross tabulation*) summarizing the relationship between the race of each fictitious job applicant and whether a callback was received. A two-way contingency table contains the number of observations that fall within each category, defined by its corresponding row (`race` variable) and column (`call` variable). Recall that a variable in a data frame can be accessed using the `$` operator (see section 1.3.5). For example, the syntax `resume$race` will extract the `race` variable in the `resume` data frame.

```
race.call.tab <- table(race = resume$race,  call = resume$call)
race.call.tab
```

```
##         call
## race       0    1
##   black 2278  157
##   white 2200  235
```

The table shows, for example, that among 2435 (= 2278 + 157) résumés with stereotypically black names, only 157 received a callback. It is convenient to add totals for each row and column by applying the `addmargins()` function to the output of the `table()` function.

```
addmargins(race.call.tab)
```

```
##         call
## race       0    1  Sum
##   black 2278  157 2435
##   white 2200  235 2435
##   Sum   4478  392 4870
```

Using this table, we can compute the callback rate, or the proportion of those who received a callback, for the entire sample and then separately for black and white applicants.

```
## overall callback rate: total callbacks divided by the sample size
sum(race.call.tab[, 2]) / nrow(resume)
```

```
## [1] 0.08049281
```

```
## callback rates for each race
race.call.tab[1, 2] / sum(race.call.tab[1, ]) # black
```

```
## [1] 0.06447639
```

```
race.call.tab[2, 2] / sum(race.call.tab[2, ]) # white
```

```
## [1] 0.09650924
```

Recall that the syntax `race.call.tab[1, ]`, which does not specify the column number, extracts all the elements of the first row of this matrix. Note that in the square brackets, the number before the comma identifies the row of the matrix whereas the number after the comma identifies the column (see section 1.3.5). This can be seen by simply typing the syntax into R.

```
race.call.tab[1, ]  # the first row
##    0    1
## 2278  157

race.call.tab[, 2]  # the second column
## black white
##   157   235
```

From this analysis, we observe that the callback rate for the résumés with African-American-sounding names is 0.032, or 3.2 percentage points, lower than those with white-sounding names. While we do not know whether this is the result of intentional discrimination, the lower callback rate for black applicants suggests the existence of racial discrimination in the labor market. Specifically, our analysis shows that the same résumé with a black-sounding name is substantially less likely to receive a callback than an identical résumé with a white-sounding name.

An easier way to compute callback rates is to exploit the fact that `call` is a *binary variable*, or *dummy variable*, that takes the value 1 if a potential employer makes a callback and 0 otherwise. In general, the sample mean of a binary variable equals the sample proportion of 1s. This means that the callback rate can be conveniently calculated as the *sample mean*, or *sample average*, of this variable using the `mean()` function rather than dividing the counts of 1s by the total number of observations. For example, instead of the slightly more complex syntax we used above, the overall callback rate can be calculated as follows.

```
mean(resume$call)
## [1] 0.08049281
```

What about the callback rate for each race? To compute this using the `mean()` function, we need to first subset the data for each race and then compute the mean of the `call` variable within this subset. The next section shows how to subset data in R.

2.2 Subsetting the Data in R

In this section, we learn how to subset a data set in various ways. We first introduce logical values and operators, which enable us to specify which observations and variables of a data set should be extracted. We also learn about factor variables, which represent categorical variables in R.

2.2.1 LOGICAL VALUES AND OPERATORS

To understand subsetting, we first note that R has a special representation of the two *logical values*, `TRUE` and `FALSE`, which belong to the object class logical (see section 1.3.2).

```
class(TRUE)
## [1] "logical"
```

These logical values can be converted to a binary variable in the integer class using the function `as.integer()`, where `TRUE` is recoded as `1` and `FALSE` becomes `0`.

```
as.integer(TRUE)
## [1] 1

as.integer(FALSE)
## [1] 0
```

In many cases, R will coerce logical values into a binary variable so that performing numerical operations is straightforward. For example, in order to compute the proportion of `TRUE`s in a vector, one can simply use the `mean()` function to compute the sample mean of a logical vector. Similarly, we can use the `sum()` function to sum the elements of this vector in order to compute the total number of `TRUE`s.

```
x <- c(TRUE, FALSE, TRUE) # a vector with logical values
mean(x) # proportion of TRUEs
## [1] 0.6666667

sum(x) # number of TRUEs
## [1] 2
```

The logical values are often produced with the *logical operators* `&` and `|` corresponding to *logical conjunction* ("*AND*") and *logical disjunction* ("*OR*"), respectively. The value of "AND" (`&`) is `TRUE` only when both of the objects have a value of `TRUE`.

```
FALSE & TRUE
## [1] FALSE

TRUE & TRUE
## [1] TRUE
```

Table 2.2. Logical Conjunction "AND" and Disjunction "OR".

Statement a	*Statement b*	*a AND b*	*a OR b*
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

The table shows the value of *a* AND *b* and that of *a* OR *b* when statements *a* and *b* are either TRUE or FALSE.

"OR" (|) is used in a similar way. However, unlike "AND", "OR" is true when at least one of the objects has the value `TRUE`.

```
TRUE | FALSE
## [1] TRUE

FALSE | FALSE
## [1] FALSE
```

We summarize these relationships in table 2.2. For example, if one statement is `FALSE` and the other is `TRUE`, then the logical conjunction of the two statements is `FALSE` but their logical disjunction is `TRUE` (the second and third rows of the table).

With the same principle in mind, we can also chain multiple comparisons together where all elements must be `TRUE` in order for the syntax to return `TRUE`.

```
TRUE & FALSE & TRUE
## [1] FALSE
```

Furthermore, "AND" and "OR" can be used simultaneously, but parentheses should be used to avoid confusion.

```
(TRUE | FALSE) & FALSE # the parentheses evaluate to TRUE
## [1] FALSE

TRUE | (FALSE & FALSE) # the parentheses evaluate to FALSE
## [1] TRUE
```

We can perform the logical operations "AND" and "OR" on the entire vector all at once. In the following syntax example, each element of the `TF1` logical vector is compared against the corresponding element of the logical `TF2` vector.

```
TF1 <- c(TRUE, FALSE, FALSE)
TF2 <- c(TRUE, FALSE, TRUE)
TF1 | TF2
## [1]  TRUE FALSE  TRUE

TF1 & TF2
## [1]  TRUE FALSE FALSE
```

2.2.2 RELATIONAL OPERATORS

Relational operators evaluate the relationships between two values. They include "greater than" (>), "greater than or equal to" (>=), "less than" (<), "less than or equal to" (<=), "equal to" (==, which is different from =), and "not equal to" (!=). These operators return logical values.

```
4 > 3
## [1] TRUE

"Hello" == "hello"  # R is case sensitive
## [1] FALSE

"Hello" != "hello"
## [1] TRUE
```

Like the logical operators, the relational operators may be applied to vectors all at once. When applied to a vector, the operators evaluate each element of the vector.

```
x <- c(3, 2, 1, -2, -1)
x >= 2
## [1]  TRUE  TRUE FALSE FALSE FALSE

x != 1
## [1]  TRUE  TRUE FALSE  TRUE  TRUE
```

Since the relational operators produce logical values, we can combine their outputs with "AND" (&) and "OR" (|). When there are multiple instances of evaluation, it is good practice to put each evaluation within parentheses for ease of interpretation.

```
## logical conjunction of two vectors with logical values
(x > 0) & (x <= 2)
## [1] FALSE  TRUE  TRUE FALSE FALSE
```

```
## logical disjunction of two vectors with logical values
(x > 2) | (x <= -1)
## [1]  TRUE FALSE FALSE  TRUE  TRUE
```

As we saw earlier, the logical values, TRUE and FALSE, can be coerced into integers (1 and 0 representing TRUE and FALSE, respectively). We can therefore compute the number and proportion of TRUE elements in a vector very easily.

```
x.int <- (x > 0) & (x <= 2) # logical vector
x.int
## [1] FALSE  TRUE  TRUE FALSE FALSE

mean(x.int) # proportion of TRUEs
## [1] 0.4

sum(x.int)  # number of TRUEs
## [1] 2
```

2.2.3 SUBSETTING

In sections 1.3.3 and 1.3.5, we learned how to subset vectors and data frames using indexing. Here, we show how to subset them using logical values, introduced above. At the end of section 2.1, we saw how to calculate the callback rate for the entire sample by applying the mean() function to the binary call variable. To compute the callback rate among the résumés with black-sounding names, we use the following syntax.

```
## callback rate for black-sounding names
mean(resume$call[resume$race == "black"])
## [1] 0.06447639
```

This command syntax subsets the call variable in the resume data frame for the observations whose values for the race variable are equal to black. That is, we can utilize square brackets [ ] to index the values in a vector by placing the logical value of each element into a vector of the same length within the square brackets. The elements whose indexing value is TRUE are extracted. The syntax then calculates the sample mean of this subsetted vector using the mean() function, which is equal to the proportion of subsetted observations whose values for the call variable are equal to 1. It is instructive to print out the logical vector used inside the square brackets for

subsetting. We observe that if the value of the `race` variable equals `black` (`white`) for an observation then its corresponding element of the resulting logical vector is `TRUE` (`FALSE`).

```
## race of first 5 observations
resume$race[1:5]

## [1] white white black black white
## Levels: black white

## comparison of first 5 observations
(resume$race == "black")[1:5]

## [1] FALSE FALSE  TRUE  TRUE FALSE
```

Note that `Levels` in the above output represent the values of a *factor* or categorical variable, which will later be explained in detail (see section 2.2.5). The calculation of callback rate for black-sounding names can also be done in two steps. We first subset a data frame object so that it contains only the résumés with black-sounding names and then compute the callback rate.

```
dim(resume) # dimension of original data frame

## [1] 4870    4

## subset blacks only
resumeB <- resume[resume$race == "black", ]
dim(resumeB) # this data.frame has fewer rows than the original data.frame

## [1] 2435    4

mean(resumeB$call) # callback rate for blacks

## [1] 0.06447639
```

Here, the data frame `resumeB` contains only the information about the résumés with black-sounding names. Notice that we used square brackets `[ , ]` to index the rows of this original data frame. Unlike in the case of indexing vectors, we use a comma to separate row and column indexes. This comma is important and forgetting to include it will lead to an error.

Instead of indexing through the square brackets, we can alternatively use the `subset()` function to construct a data frame that contains just some of the original observations and just some of the original variables. The function's two primary arguments, other than the original data frame object, are the `subset` and `select` arguments. The `subset` argument takes a logical vector that indicates whether each individual row should be kept for the new data frame. The `select` argument takes a character vector that specifies the names of variables to be retained. For example,

the following syntax will extract the `call` and `firstname` variables for the résumés which contain female black-sounding names.

```
## keep "call" and "firstname" variables
## also keep observations with female black-sounding names
resumeBf <- subset(resume, select = c("call", "firstname"),
                   subset = (race == "black" & sex == "female"))
head(resumeBf)
##    call firstname
## 3     0   Lakisha
## 4     0   Latonya
## 8     0     Kenya
## 9     0   Latonya
## 11    0     Aisha
## 13    0     Aisha
```

When using the `subset()` function, we can eliminate the `subset` argument label. For example, `subset(resume, subset = (race == "black" & sex == "female"))` shortens to `subset(resume, race == "black" & sex == "female")`. Note that one could specify the data frame name to which the `race` and `sex` variables belong, i.e., `subset(resume, (resume$race == "black" & resume$sex == "female"))`, but this is unnecessary. By default, the variable names in this argument are assumed to come from the data frame specified in the first argument (`resume` in this case). So we can use simpler syntax: `subset(resume, (race == "black" & sex == "female"))`. It is important to pay close attention to parentheses so that each logical statement is contained within a pair of parentheses.

An identical subsetting result can be obtained using `[,]` rather than the `subset()` function, where the first element of the square brackets specifies the rows to be retained (using a logical vector) and the second element specifies the columns to be kept (using a character or integer vector).

```
## alternative syntax with the same results
resumeBf <- resume[resume$race == "black" & resume$sex == "female",
                   c("call", "firstname")]
```

We can now separately compute the racial gap in callback rate among female and male job applicants. Notice that we do not include a `select` argument to specify which variables to keep. Consequently, all variables will be retained.

```
## black male
resumeBm <- subset(resume, subset = (race == "black") & (sex == "male"))
```

```
## white female
resumeWf <- subset(resume, subset = (race == "white") & (sex == "female"))
## white male
resumeWm <- subset(resume, subset = (race == "white") & (sex == "male"))
## racial gaps
mean(resumeWf$call) - mean(resumeBf$call) # among females
## [1] 0.03264689

mean(resumeWm$call) - mean(resumeBm$call) # among males
## [1] 0.03040786
```

It appears that the racial gap exists but does not vary across gender groups. For both female and male job applicants, the callback rate is higher for whites than blacks by roughly 3 percentage points.

2.2.4 SIMPLE CONDITIONAL STATEMENTS

In many situations, we would like to perform different actions depending on whether a statement is true or false. These "actions" can be as complex or as simple as you need them to be. For example, we may wish to create a new variable based on the values of other variables in a data set. In chapter 4, we will learn more about *conditional statements*, but here we cover simple conditional statements that involve the `ifelse()` function.

The function `ifelse(X, Y, Z)` contains three elements. For each element in `X` that is `TRUE`, the corresponding element in `Y` is returned. In contrast, for each element in `X` that is `FALSE`, the corresponding element in `Z` is returned. For example, suppose that we want to create a new binary variable called `BlackFemale` in the `resume` data frame that equals `1` if the job applicant's name sounds black and female, and `0` otherwise. The following syntax achieves this goal.

```
resume$BlackFemale <- ifelse(resume$race == "black" &
                             resume$sex == "female", 1, 0)
```

We then use a three-way *contingency table* obtained by the `table()` function to confirm the result. As expected, the `BlackFemale` variable equals `1` only when a résumé belongs to a female African-American.

```
table(race = resume$race, sex = resume$sex,
      BlackFemale = resume$BlackFemale)
## , , BlackFemale = 0
##
##         sex
## race    female male
##   black      0  549
##   white   1860  575
```

```
##
## , , BlackFemale = 1
##
##         sex
## race    female male
##   black   1886    0
##   white      0    0
```

In the above output, the `, , BlackFemale = 0` and `, , BlackFemale = 1` headers indicate that the first two dimensions of the three-dimensional table are shown with the third variable, `BlackFemale`, equal to `0` and `1` for the first and second tables, respectively.

2.2.5 FACTOR VARIABLES

Next we show how to create a *factor variable* (or *factorial variable*) in R. A factor variable is another name for a *categorical variable* that takes a finite number of distinct values or levels. Here, we wish to create a factor variable that takes one of the four values, i.e., `BlackFemale`, `BlackMale`, `WhiteFemale`, and `WhiteMale`. To do this, we first create a new variable, `type`, which is filled with missing values `NA`. We then specify each type using the characteristics of the applicants.

```
resume$type <- NA
resume$type[resume$race == "black" & resume$sex == "female"] <- "BlackFemale"
resume$type[resume$race == "black" & resume$sex == "male"] <- "BlackMale"
resume$type[resume$race == "white" & resume$sex == "female"] <- "WhiteFemale"
resume$type[resume$race == "white" & resume$sex == "male"] <- "WhiteMale"
```

It turns out that this new variable is a character vector, and so we use the `as.factor()` function to turn this vector into a factor variable. While a factor variable looks like a character variable, the former actually has numeric values called *levels*, each of which has a character label. By default, the levels are sorted into alphabetical order based on their character labels. The levels of a factor variable can be obtained using the `levels()` function. Moreover, the `table()` function can be applied to obtain the number of observations that fall into each level.

```
## check object class
class(resume$type)
## [1] "character"

## coerce new character variable into a factor variable
resume$type <- as.factor(resume$type)
## list all levels of a factor variable
levels(resume$type)
## [1] "BlackFemale" "BlackMale" "WhiteFemale" "WhiteMale"
```

```
## obtain the number of observations for each level
table(resume$type)

##
## BlackFemale   BlackMale WhiteFemale   WhiteMale
##        1886         549        1860         575
```

The main advantage of factor objects is that R has a number of useful functionalities for them. One such example is the `tapply()` function, which applies a function repeatedly within each level of the factor variable. Suppose, for example, we want to calculate the callback rate for each of the four categories we just created. If we use the `tapply()` function this can be done in one line, rather than computing them one by one. Specifically, we use the function as in `tapply(X, INDEX, FUN)`, which applies the function indicated by argument `FUN` to the object `X` for each of the groups defined by unique values of the vector `INDEX`. Here, we apply the `mean()` function to the `call` variable separately for each category of the `type` variable using the `resume` data frame.

```
tapply(resume$call, resume$type, mean)

## BlackFemale   BlackMale WhiteFemale   WhiteMale
##  0.06627784  0.05828780  0.09892473  0.08869565
```

Recall that the order of arguments in a function matters unless the name of the argument is explicitly specified. The result indicates that black males have the lowest callback rate followed by black females, white males, and white females. We can even go one step further and compute the callback rate for each first name. Using the `sort()` function, we can sort the result into increasing order for ease of presentation.

```
## turn first name into a factor variable
resume$firstname <- as.factor(resume$firstname)
## compute callback rate for each first name
callback.name <- tapply(resume$call, resume$firstname, mean)
## sort the result into increasing order
sort(callback.name)

##      Aisha    Rasheed     Keisha   Tremayne     Kareem
## 0.02222222 0.02985075 0.03825137 0.04347826 0.04687500
##    Darnell     Tyrone      Hakim     Tamika    Lakisha
## 0.04761905 0.05333333 0.05454545 0.05468750 0.05500000
##    Tanisha       Todd      Jamal       Neil      Brett
## 0.05797101 0.05882353 0.06557377 0.06578947 0.06779661
##   Geoffrey    Brendan       Greg      Emily       Anne
## 0.06779661 0.07692308 0.07843137 0.07929515 0.08264463
```

```
##       Jill     Latoya      Kenya    Matthew    Latonya
## 0.08374384 0.08407080 0.08673469 0.08955224 0.09130435
##      Leroy    Allison      Ebony   Jermaine     Laurie
## 0.09375000 0.09482759 0.09615385 0.09615385 0.09743590
##      Sarah   Meredith     Carrie    Kristen        Jay
## 0.09844560 0.10160428 0.13095238 0.13145540 0.13432836
##       Brad
## 0.15873016
```

As expected from the above aggregate result, we find that many typical names for black males and females have low callback rates.

2.3 Causal Effects and the Counterfactual

In the résumé experiment, we are trying to quantify the *causal effects* of applicants' names on their likelihood of receiving a callback from a potential employer. What do we exactly mean by causal effects? How should we think about causality in general? In this section, we discuss a commonly used framework for *causal inference* in quantitative social science research.

The key to understanding causality is to think about the *counterfactual*. Causal inference is a comparison between the factual (i.e., what actually happened) and the counterfactual (i.e., what would have happened if a key condition were different). The very first observation of the résumé experiment data shows that a potential employer received a résumé with a stereotypically white female first name `Allison` but decided not to call back (the value of the `call` variable is `0` for this observation).

```
resume[1, ]
##   firstname    sex  race call BlackFemale        type
## 1   Allison female white    0           0 WhiteFemale
```

The key causal question here is whether the same employer would have called back if the applicant's name were instead a stereotypically African-American name such as `Lakisha`. Unfortunately, we would never observe this counterfactual outcome, because the researchers who conducted this experiment did not send out the same résumé to the same employer using `Lakisha` as the first name (perhaps out of fear that sending two identical résumés with different names would raise suspicion among potential employers).

Consider another example where researchers are interested in figuring out whether raising the minimum wage increases the unemployment rate. Some argue that increasing the minimum wage may not be helpful for the poor, because employers would hire fewer workers if they have to pay higher wages (or hire higher-skilled instead of low-skilled workers). Suppose that one state in a country decided to raise the minimum wage and in this state the unemployment rate increased afterwards. This does not

Table 2.3. Potential Outcome Framework of Causal Inference.

Résumé i	*Black-sounding name* T_i	*Callback* $Y_i(1)$	*Callback* $Y_i(0)$	*Age*	*Education*
1	1	1	?	20	college
2	0	?	0	55	high school
3	0	?	1	40	graduate school
⋮	⋮	⋮	⋮	⋮	⋮
n	1	0	?	62	college

Note: The table illustrates the potential outcome framework of causal inference using the example of the résumé experiment. For each résumé of fictitious job applicant i, either the black-sounding, $T_i = 1$, or white-sounding, $T_i = 0$, name is used. The résumé contains other characteristics such as age and education, which are neither subject to nor affected by the manipulation. For a résumé with a black-sounding name, we can observe whether or not it receives a callback from the potential employer who received it, $Y_i(1)$, but will not be able to know the callback outcome if a white-sounding name was used, $Y_i(0)$. For every résumé, only one of the two potential outcomes is observed and the other is missing (indicated by "?").

necessarily imply that a higher minimum wage led to the increase in the unemployment rate. In order to know the causal effect of increasing the minimum wage, we would need to observe the unemployment rate that would have resulted if this state had not raised the minimum wage. Clearly, we would never be able to directly survey this counterfactual unemployment rate. Another example concerns the question of whether a job training program increases one's prospect of employment. Even if someone who actually had received job training secured a job afterwards, it does not necessarily follow that it was the job training program which led to the employment. The person may have become employed even in the absence of such a training program.

These examples illustrate the *fundamental problem of causal inference*, which arises because we cannot observe the counterfactual outcomes. We refer to a key causal variable of interest as a *treatment variable*, even though the variable may have nothing to do with a medical treatment. To determine whether a *treatment* variable of interest T, causes a change in an outcome variable Y, we must consider two *potential outcomes*, i.e., the potential values of Y that would be realized in the presence and absence of the treatment, denoted by $Y(1)$ and $Y(0)$, respectively. In the résumé experiment, T may represent the race of a fictitious applicant ($T = 1$ is a black-sounding name and $T = 0$ is a white-sounding name) while Y denotes whether a potential employer who received the résumé called back. Then, $Y(1)$ and $Y(0)$ represent whether a potential employer calls back when receiving a résumé with stereotypically black and white names, respectively.

All of these variables can be defined for each observation and marked by a corresponding subscript. For example, $Y_i(1)$ represents the potential outcome under the treatment condition for the ith observation, and T_i is the treatment variable for the same observation. Table 2.3 illustrates the potential outcome framework in the context of the résumé experiment. Each row represents an observation for which only one of the two potential outcomes is observed (the missing potential outcome is indicated by "?"). The treatment status T_i determines which potential outcome is observed. Variables such as age and education are neither subject to nor affected by the manipulation of treatment.

We can now define, for each observation, the causal effect of T_i on Y_i as the difference between these two potential outcomes, $Y_i(1) - Y_i(0)$. The race of the applicant has a causal effect if a potential employer's decision to callback depends on it. As stated earlier, the fundamental problem of causal inference is that we are only able to observe one of the two potential outcomes even though causal inference requires comparison of both. An important implication is that for estimation of causal effects, we must find a credible way to infer these unobserved counterfactual outcomes. This requires making certain assumptions. The credibility of any causal inference, therefore, rests upon the plausibility of these identification assumptions.

For each observation i, we can define the **causal effect** of a binary treatment T_i as the difference between two potential outcomes, $Y_i(1) - Y_i(0)$, where $Y_i(1)$ represents the outcome that would be realized under the treatment condition ($T_i = 1$) and $Y_i(0)$ denotes the outcome that would be realized under the control condition ($T_i = 0$).

The **fundamental problem of causal inference** is that we observe only one of the two potential outcomes, and which potential outcome is observed depends on the treatment status. Formally, the observed outcome Y_i is equal to $Y_i(T_i)$.

This simple framework of causal inference also clarifies what is and is not an appropriate causal question. For example, consider a question of whether one's race causally affects one's employment prospects. In order to answer this question directly, it would be necessary to consider the counterfactual employment status if the applicant were to belong to a different racial group. However, this is a difficult proposition to address because one's race is not something that can be manipulated. Characteristics like gender and race are called *immutable characteristics*, and many scholars believe that causal questions about these characteristics are not answerable. In fact, there exists a mantra which states, "No causation without manipulation." It may be difficult to think about causality if the treatment variable of interest cannot be easily manipulated.

The résumé experiment, however, provides a clever way of addressing an important social science question about racial discrimination. Instead of tackling the difficult task of directly estimating the causal effect of race, the researchers of this study manipulated potential employers' *perception* of job applicants' race by changing the names on identical résumés. This research design strategy enables one to study racial discrimination in the causal inference framework by circumventing the difficulty of manipulating one's race itself. Many social scientists use similar research design strategies to study discrimination due to factors such as race, gender, and religion in various environments.

2.4 Randomized Controlled Trials

Now that we have provided the general definition of causal effects, how should we go about estimating them? We first consider *randomized experiments*, also referred

to as *randomized controlled trials* (RCTs), in which researchers randomly assign the receipt of treatment. An RCT is often regarded as the gold standard for establishing causality in many scientific disciplines because it enables researchers to isolate the effects of a treatment variable and quantify uncertainty. In this section, we discuss how randomization identifies the average causal effects. A discussion of how to quantify uncertainty will be given in chapter 7.

2.4.1 THE ROLE OF RANDOMIZATION

As explained in the previous section, the fundamental problem of causal inference states that for the estimation of causal effects, we must infer counterfactual outcomes. This problem prevents us from obtaining a valid estimate of the causal effect of treatment for each individual. However, it turns out that the randomization of treatment assignment enables the estimation of *average treatment effect*, which averages the treatment effect over a group of individuals.

Suppose that we are interested in estimating the *sample average treatment effect* (SATE), which is defined as the average of individual-level treatment effects in the sample.

The **sample average treatment effect** (SATE) is defined as the sample average of individual-level causal effects (i.e., $Y_i(1) - Y_i(0)$):

$$\text{SATE} = \frac{1}{n}\sum_{i=1}^{n}\{Y_i(1) - Y_i(0)\}, \tag{2.1}$$

where n is the sample size, and $\sum_{i=1}^{n}$ denotes the summation operator from the first observation, $i = 1$, to the last, $i = n$.

The SATE is not directly observable. For the *treatment group* that received the treatment, we observe the average outcome under the treatment but do not know what their average outcome would have been in the absence of the treatment. The same problem exists for the *control group* because this group does not receive the treatment and as a result we do not observe the average outcome that would occur under the treatment condition.

In order to estimate the average counterfactual outcome for the treatment group, we may use the observed average outcome of the control group. Similarly, we can use the observed average outcome of the treatment group as an estimate of the average counterfactual outcome for the control group. This suggests that the SATE can be estimated by calculating the difference in the average outcome between the treatment and control groups or the *difference-in-means estimator*. The critical question is whether we can interpret this difference as a valid estimate of the average causal effect. In the résumé experiment, the treatment group consists of the potential employers who were sent résumés with black-sounding names. In contrast, the control group comprises other potential employers who received the résumés with stereotypically white names. Does the difference in callback rate between these two groups represent the average causal effect of the applicant's race?

Randomization of treatment assignment plays an essential role in enabling the interpretation of this *association* as a causal relationship. By randomly assigning each subject to either the treatment or control group, we ensure that these two groups are similar to each other in every aspect. In fact, even though they consist of different individuals, the treatment and control groups are *on average* identical to each other in terms of *all* pretreatment characteristics, both observed and unobserved. Since the only systematic difference between the two groups is the receipt of treatment, we can interpret the difference in the outcome variable as the estimated average causal effect of the treatment. In this way, the randomization of treatment assignment separates the causal effect of treatment from other possible factors that may influence the outcome. As we will see in section 2.5, we cannot guarantee that the treatment and control groups are comparable across all unobserved characteristics in the absence of random assignment.

In a **randomized controlled trial (RCT)**, each unit is randomly assigned either to the treatment or control group. The randomization of treatment assignment guarantees that the average difference in outcome between the treatment and control groups can be attributed solely to the treatment, because the two groups are on average identical to each other in all pretreatment characteristics.

RCTs, when successfully implemented, can yield valid estimates of causal effects. For this reason, RCTs are said to have a significant advantage for *internal validity*, which refers to whether the causal assumptions are satisfied in the study. However, RCTs are not without weaknesses. In particular, their strong internal validity often comes with a compromise in *external validity*. External validity is defined as the extent to which the conclusions can be generalized beyond a particular study. One common reason for a lack of external validity is that the study sample may not be representative of a population of interest. For ethical and logistical reasons, RCTs are often done using a convenient sample of subjects who are willing to be study subjects. This is an example of *sample selection bias*, making the experimental sample nonrepresentative of a target population. Another potential problem of external validity is that RCTs are often conducted in an environment (e.g., laboratory) quite different from real-world situations. In addition, RCTs may use interventions that are unrealistic in nature. As we saw in the résumé experiment, however, researchers have attempted to overcome these problems by conducting RCTs in the field and making their interventions as realistic as possible.

The main advantage of randomized controlled trials (RCTs) is their improved **internal validity**—the extent to which causal assumptions are satisfied in the study. One weakness of RCTs, however, is the potential lack of **external validity**—the extent to which the conclusions can be generalized beyond a particular study.

Dear Registered Voter:

WHAT IF YOUR NEIGHBORS KNEW WHETHER YOU VOTED?

Why do so many people fail to vote? We've been talking about the problem for years, but it only seems to get worse. This year, we're taking a new approach.
We're sending this mailing to you and your neighbors to publicize who does and does not vote.

The chart shows the names of some of your neighbors, showing which have voted in the past. After the August 8 election, we intend to mail an updated chart. You and your neighbors will all know who voted and who did not.

DO YOUR CIVIC DUTY – VOTE!

MAPLE DR	Aug 04	Nov 04	Aug 06
9995 JOSEPH JAMES SMITH	Voted	Voted	_____
995 JENNIFER KAY SMITH		Voted	_____
9997 RICHARD B JACKSON		Voted	_____
9999 KATHY MARIE JACKSON		Voted	_____

Figure 2.1. Naming-and-Shaming Get-out-the-Vote Message. Reprinted from Gerber, Green, and Larimer (2008).

2.4.2 SOCIAL PRESSURE AND VOTER TURNOUT

We consider a study of peer pressure and voter turnout,[2] another example of an RCT. Three social scientists conducted an RCT in which they investigated whether social pressure within neighborhoods increases participation. Specifically, during a primary election in the state of Michigan, they randomly assigned registered voters to receive different *get-out-the-vote* (GOTV) messages and examined whether sending postcards with these messages increased turnout. The researchers exploited the fact that the turnout of individual voters is public information in the United States.

The GOTV message of particular interest was designed to induce social pressure by telling voters that after the election their neighbors would be informed about whether they voted in the election or not. The researchers hypothesized that such a naming-and-shaming GOTV strategy would increase participation. An example of the actual naming-and-shaming message is shown in figure 2.1. In addition to the control group, which did not receive any mailing, the study also included other GOTV messages. For example, a standard "civic duty" message began with the same first two sentences of the naming-and-shaming message, but did not contain the additional information about neighbors learning about a person's electoral participation. Instead, the message continued to read as follows:

> The whole point of democracy is that citizens are active participants in government; that we have a voice in government. Your voice starts with your vote. On August 8, remember your rights and responsibilities as a citizen. Remember to vote. DO YOUR CIVIC DUTY – VOTE!

[2] This section is based on Alan S. Gerber, Donald P. Green, and Christopher W. Larimer (2008) "Social pressure and voter turnout: Evidence from a large-scale field experiment." *American Political Science Review*, vol. 102, no. 1, pp. 33–48.

Another important feature of this RCT is that the researchers attempted to separate the effect of naming-and-shaming from that of being observed. In many RCTs, there is a concern that study subjects may behave differently if they are aware of being observed by researchers. This phenomenon is called the *Hawthorne effect*, named after the factory where researchers observed an increase in workers' productivity simply because they knew that they were being monitored as part of a study. To address this issue, the study included another GOTV message, which starts with "YOU ARE BEING STUDIED!" followed by the same first two sentences as the naming-and-shaming message. The rest of the message reads,

> This year, we're trying to figure out why people do or do not vote. We'll be studying voter turnout in the August 8 primary election. Our analysis will be based on public records, so you will not be contacted again or disturbed in any way. Anything we learn about your voting or not voting will remain confidential and will not be disclosed to anyone else. DO YOUR CIVIC DUTY – VOTE!

The **Hawthorne effect** refers to the phenomenon where study subjects behave differently because they know they are being observed by researchers.

In this experiment, therefore, there are three treatment groups: voters who receive either the social pressure message, the civic duty message, or the Hawthorne effect message. The experiment also has a control group which consists of those voters receiving no message. The researchers randomly assigned each voter to one of the four groups and examined whether the voter turnout was different across the groups.

Now that we understand the design of this experiment, let us analyze the data. The data file, which is in CSV format, is named `social.csv` and can be loaded into R via the `read.csv()` function. Table 2.4 displays the names and descriptions of the variables in the social pressure experiment data.

```
social <- read.csv("social.csv") # load the data
summary(social) # summarize the data
##      sex          yearofbirth    primary2004
##  female:152702   Min.   :1900   Min.   :0.0000
##  male  :153164   1st Qu.:1947   1st Qu.:0.0000
##                  Median :1956   Median :0.0000
##                  Mean   :1956   Mean   :0.4014
##                  3rd Qu.:1965   3rd Qu.:1.0000
##                  Max.   :1986   Max.   :1.0000
##        messages       primary2006         hhsize
##  Civic Duty: 38218   Min.   :0.0000   Min.   :1.000
##  Control   :191243   1st Qu.:0.0000   1st Qu.:2.000
##  Hawthorne : 38204   Median :0.0000   Median :2.000
##  Neighbors : 38201   Mean   :0.3122   Mean   :2.184
##                      3rd Qu.:1.0000   3rd Qu.:2.000
##                      Max.   :1.0000   Max.   :8.000
```

Table 2.4. Social Pressure Experiment Data.

Variable	*Description*
`hhsize`	household size of the voter
`messages`	GOTV messages the voter received (`Civic Duty`, `Control`, `Neighbors`, `Hawthorne`)
`sex`	sex of the voter (`female` or `male`)
`yearofbirth`	year of birth of the voter
`primary2004`	whether the voter voted in the 2004 primary election (`1`=voted, `0`=abstained)
`primary2006`	whether the voter turned out in the 2006 primary election (`1`=voted, `0`=abstained)

As shown in section 2.2.5, we can use the `tapply()` function to compute the turnout for each treatment group. Subtracting the baseline turnout from the control group gives the average causal effect of each message. Note that the outcome variable of interest is the turnout in the 2006 primary election, which is coded as a binary variable `primary2006` where `1` represents turnout and `0` is abstention.

```
## turnout for each group
tapply(social$primary2006, social$messages, mean)

## Civic Duty    Control  Hawthorne  Neighbors
##  0.3145377  0.2966383  0.3223746  0.3779482

## turnout for control group
mean(social$primary2006[social$messages == "Control"])

## [1] 0.2966383

## subtract control group turnout from each group
tapply(social$primary2006, social$messages, mean) -
    mean(social$primary2006[social$messages == "Control"])

## Civic Duty    Control  Hawthorne  Neighbors
## 0.01789934 0.00000000 0.02573631 0.08130991
```

We find that the naming-and-shaming GOTV message substantially increases turnout. Compared to the control group turnout, the naming-and-shaming message increases turnout by 8.1 percentage points, whereas the civic duty message has a much smaller effect of 1.8 percentage points. It is interesting to see that the *Hawthorne effect* of being observed is somewhat greater than the effect of the civic duty message, though it is far smaller than the effect of the naming-and-shaming message.

Finally, if the randomization of treatment assignment is successful, we should not observe large differences across groups in the *pretreatment variables* such as age (indicated by `yearofbirth`), turnout in the previous primary election (`primary2004`), and household size (`hhsize`). We examine these using the same syntax.

```
social$age <- 2006 - social$yearofbirth # create age variable
tapply(social$age, social$messages, mean)
## Civic Duty    Control  Hawthorne  Neighbors
##   49.65904   49.81355   49.70480   49.85294

tapply(social$primary2004, social$messages, mean)
## Civic Duty    Control  Hawthorne  Neighbors
##  0.3994453  0.4003388  0.4032300  0.4066647

tapply(social$hhsize, social$messages, mean)
## Civic Duty    Control  Hawthorne  Neighbors
##   2.189126   2.183667   2.180138   2.187770
```

We see that the differences in these pretreatment variables are negligible across groups, confirming that the randomization of treatment assignment makes the four groups essentially identical to one another on average.

2.5 Observational Studies

Although RCTs can provide an internally valid estimate of causal effects, in many cases social scientists are unable to randomize treatment assignment in the real world for ethical and logistical reasons. We next consider *observational studies* in which researchers do not conduct an intervention. Instead, in observational studies, researchers simply observe naturally occurring events and collect and analyze the data. In such studies, *internal validity* is likely to be compromised because of possible selection bias, but *external validity* is often stronger than that of RCTs. The findings from observational studies are typically more generalizable because researchers can examine the treatments that are implemented among a relevant population in a real-world environment.

2.5.1 MINIMUM WAGE AND UNEMPLOYMENT

Our discussion of observational studies is based on the aforementioned minimum-wage debate. Two social science researchers examined the impact of raising the minimum wage on employment in the fast-food industry.[3] In 1992, the state of New Jersey (NJ) in the United States raised the minimum wage from \$4.25 to \$5.05 per hour. Did such an increase in the minimum wage reduce employment as economic theory predicts? As discussed above, answering this question requires inference about the NJ employment rate in the absence of such a raise in the minimum wage. Since this

[3] This section is based on David Card and Alan Krueger (1994) "Minimum wages and employment: A case study of the fast-food industry in New Jersey and Pennsylvania." *American Economic Review*, vol. 84, no. 4, pp. 772–793.

Table 2.5. Minimum-Wage Study Data.

Variable	*Description*
`chain`	name of the fast-food restaurant chain
`location`	location of the restaurants (`centralNJ`, `northNJ`, `PA`, `shoreNJ`, `southNJ`)
`wageBefore`	wage before the minimum-wage increase
`wageAfter`	wage after the minimum-wage increase
`fullBefore`	number of full-time employees before the minimum-wage increase
`fullAfter`	number of full-time employees after the minimum-wage increase
`partBefore`	number of part-time employees before the minimum-wage increase
`partAfter`	number of part-time employees after the minimum-wage increase

counterfactual outcome is not observable, we must somehow estimate it using observed data.

One possible strategy is to look at another state in which the minimum wage did not increase. For example, the researchers of this study chose the neighboring state, Pennsylvania (PA), on the grounds that NJ's economy resembles that of Pennsylvania, and hence the fast-food restaurants in the two states are comparable. Under this *cross-section comparison design*, therefore, the fast-food restaurants in NJ serve as the *treatment group* receiving the treatment (i.e., the increase in the minimum wage), whereas those in PA represent the *control group*, which did not receive such a treatment. To collect pretreatment and outcome measures, the researchers surveyed the fast-food restaurants before and after the minimum wage increase. Specifically, they gathered information about the number of full-time employees, the number of part-time employees, and their hourly wages, for each restaurant.

The CSV file `minwage.csv` contains this data set. As usual, the `read.csv()` function loads the data set, the `dim()` function gives the number of observations and the number of variables, and the `summary()` function provides a summary of each variable. Table 2.5 displays the names and descriptions of the variables in the minimum-wage study data.

```
minwage <- read.csv("minwage.csv") # load the data
dim(minwage) # dimension of data
## [1] 358   8

summary(minwage) # summary of data
##         chain          location     wageBefore
##  burgerking:149   centralNJ: 45   Min.   :4.250
##  kfc        : 75   northNJ  :146   1st Qu.:4.250
```

```
##  roys      : 88   PA       : 67   Median :4.500
##  wendys    : 46   shoreNJ  : 33   Mean   :4.618
##                   southNJ  : 67   3rd Qu.:4.987
##                                    Max.   :5.750
##    wageAfter       fullBefore       fullAfter
##  Min.   :4.250   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:5.050   1st Qu.: 2.125   1st Qu.: 2.000
##  Median :5.050   Median : 6.000   Median : 6.000
##  Mean   :4.994   Mean   : 8.475   Mean   : 8.362
##  3rd Qu.:5.050   3rd Qu.:12.000   3rd Qu.:12.000
##  Max.   :6.250   Max.   :60.000   Max.   :40.000
##    partBefore      partAfter
##  Min.   : 0.00   Min.   : 0.00
##  1st Qu.:11.00   1st Qu.:11.00
##  Median :16.25   Median :17.00
##  Mean   :18.75   Mean   :18.69
##  3rd Qu.:25.00   3rd Qu.:25.00
##  Max.   :60.00   Max.   :60.00
```

To make sure that the restaurants followed the law, we first examine whether the minimum-wage actually increased in NJ after the law was enacted. We first subset the data based on location and then calculate the proportion of restaurants in each state with hourly wages less than the new minimum wage in NJ, i.e., \$5.05. This analysis can be done using the `wageBefore` and `wageAfter` variables, which represent the wage before and after the NJ law went into effect. The `subset()` function can be used to conduct this analysis.

```
## subsetting the data into two states
minwageNJ <- subset(minwage, subset = (location != "PA"))
minwagePA <- subset(minwage, subset = (location == "PA"))
## proportion of restaurants whose wage is less than $5.05
mean(minwageNJ$wageBefore < 5.05) # NJ before
## [1] 0.9106529

mean(minwageNJ$wageAfter < 5.05)  # NJ after
## [1] 0.003436426

mean(minwagePA$wageBefore < 5.05) # PA before
## [1] 0.9402985

mean(minwagePA$wageAfter < 5.05)  # PA after
## [1] 0.9552239
```

We observe that more than 91% of NJ restaurants were paying less than \$5.05 before the minimum wage was raised and yet afterwards the proportion of such restaurants dramatically declined to less than 1%. In contrast, this proportion is essentially unchanged in PA, suggesting that the NJ law had minimal impact on the wages in PA restaurants. The analysis shows that the NJ restaurants followed the law by increasing their wage above the new minimum wage \$5.05 while the PA restaurants did not have to make a similar change.

We now use the PA restaurants as the control group and estimate the average causal effect of increasing the minimum wage on employment among the NJ restaurants. An economic theory would predict that raising the minimum wage will encourage employers to replace full-time employees with part-time ones to recoup the increased cost in wages. To test this theory, we examine the proportion of full-time employees as a key outcome variable by simply comparing the *sample mean* of this variable between the NJ and PA restaurants after the NJ law went into effect. Let's compute this difference-in-means estimator.

```
## create a variable for proportion of full-time employees in NJ and PA
minwageNJ$fullPropAfter <- minwageNJ$fullAfter /
      (minwageNJ$fullAfter + minwageNJ$partAfter)
minwagePA$fullPropAfter <- minwagePA$fullAfter /
    (minwagePA$fullAfter + minwagePA$partAfter)
## compute the difference-in-means
mean(minwageNJ$fullPropAfter) - mean(minwagePA$fullPropAfter)
## [1] 0.04811886
```

The result of this analysis suggests that the increase in the minimum wage had no negative impact on employment. If anything, it appears to have slightly increased the proportion of full-time employment in NJ fast-food restaurants.

2.5.2 CONFOUNDING BIAS

The important assumption of observational studies is that the treatment and control groups must be comparable with respect to everything related to the outcome other than the treatment. In the current example, we cannot attribute the above difference in the full-time employment rate between NJ and PA restaurants to the minimum-wage increase in NJ if, for example, there is a competing industry for low-skilled workers in NJ but such an industry does not exist in PA. If that is the case, then the restaurants in the two states are not comparable and PA restaurants cannot serve as a valid control group for NJ restaurants. Indeed, NJ restaurants may have had a relatively high full-time employment rate, even in the absence of the increased minimum wage, in order to attract low-skilled workers. More generally, any other differences that exist between the fast-food restaurants in the two states before the administration of the NJ law would bias our inference if they are also related to outcomes.

The *pretreatment variables* that are associated with both the treatment and outcome variables are known as *confounders*. They are the variables that are realized prior to the administration of treatment and hence are not causally affected by the treatment.

However, they may determine who is likely to receive the treatment and influence the outcome. The existence of such variables is said to confound the causal relationship between the treatment and outcome, making it impossible to draw causal inferences from observational data. *Confounding bias* of this type is often a serious concern for social science research because in many cases human beings self-select into treatments. The aforementioned possibility that there exists a competing industry in NJ but not in PA is an example of confounding.

A pretreatment variable that is associated with both the treatment and the outcome variables is called a **confounder** and is a source of **confounding bias** in the estimation of the treatment effect.

Confounding bias due to self-selection into the treatment group is called *selection bias*. Selection bias often arises in observational studies because researchers have no control over who receives the treatment. In the minimum-wage study, NJ politicians decided to increase the minimum wage at this particular moment in time whereas politicians in PA did not. One might suspect that there were reasons, related to the economy and employment in particular, why the minimum wage was raised in NJ but not in PA. If that is the case, then the cross-sectional comparison of NJ and PA after the minimum-wage increase in NJ is likely to yield selection bias. The lack of control over treatment assignment means that those who self-select themselves into the treatment group may differ significantly from those who do not in terms of observed and unobserved characteristics. This makes it difficult to determine whether the observed difference in outcome between the treatment and control groups is due to the difference in the treatment condition or the differences in confounders. The possible existence of confounding bias is the reason behind the existence of the popular mantra, "Association does not necessarily imply causation."

In observational studies, the possibility of confounding bias can never be ruled out. However, researchers can try to address it by means of *statistical control*, whereby the researcher adjusts for confounders using statistical procedures. We describe some basic strategies in this section. One simple way is the statistical method called *subclassification*. The idea is to make the treatment and control groups as similar to each other as possible by comparing them within a subset of observations defined by shared values in pretreatment variables or a subclass. For example, we notice that the PA sample has a larger proportion of Burger Kings than the NJ sample. This difference between the two states could confound the relationship between minimum-wage increase and employment if, for example, Burger King has an employment policy that is different from that of other fast-food chains. To address this possibility, we could conduct a comparison only among Burger King restaurants. This analysis enables us to eliminate the confounding bias due to different fast-food chains through statistical control.

To begin our analysis, we first check the proportions of different fast-food chains for each of the two samples. We use the `prop.table()` function, which takes as its main input the output from the `table()` function, i.e., a table of counts, and converts it to proportions.

```
prop.table(table(minwageNJ$chain))

##
## burgerking        kfc       roys     wendys
##  0.4054983  0.2233677  0.2508591  0.1202749

prop.table(table(minwagePA$chain))

##
## burgerking        kfc       roys     wendys
##  0.4626866  0.1492537  0.2238806  0.1641791
```

The result shows that PA has a higher proportion of Burger King restaurants than NJ. We compare the full-time employment rate between NJ and PA Burger King restaurants after the increase in the minimum wage. Though not shown here, a similar analysis can be conducted for other fast-food chain restaurants as well.

```
## subset Burger King only
minwageNJ.bk <- subset(minwageNJ, subset = (chain == "burgerking"))
minwagePA.bk <- subset(minwagePA, subset = (chain == "burgerking"))
## comparison of full-time employment rates
mean(minwageNJ.bk$fullPropAfter) - mean(minwagePA.bk$fullPropAfter)

## [1] 0.03643934
```

This finding is quite similar to the overall result presented earlier, suggesting that the fast-food chain may not be a confounding factor.

Another possible confounder is the location of restaurants. In particular, it may be the case that the NJ Burger King restaurants closer to PA yield a more credible comparison to those in PA, perhaps because their local economies share similar characteristics. To address this possible confounding bias, we may further subclassify the data on the basis of restaurant location. Specifically, we focus on the Burger King restaurants located in northern and southern NJ that are near PA, while excluding those in the Jersey shore and central New Jersey, and repeat the analysis. This analysis adjusts for both the type of restaurants and their locations through statistical control.

```
minwageNJ.bk.subset <-
    subset(minwageNJ.bk, subset = ((location != "shoreNJ") &
                                   (location != "centralNJ")))
mean(minwageNJ.bk.subset$fullPropAfter) - mean(minwagePA.bk$fullPropAfter)

## [1] 0.03149853
```

The result shows that even within this smaller subset of the original data, the estimated impact of the minimum-wage increase remains similar to the overall estimate. This finding further improves our confidence in the claim that the increase in the minimum wage had little effect on full-time employment.

Confounding bias can be reduced through **statistical control**. For example, we can use the method of **subclassification** by comparing treated and control units which have an identical value of a confounding variable.

2.5.3 BEFORE-AND-AFTER AND DIFFERENCE-IN-DIFFERENCES DESIGNS

In observational studies, the data collected over time are a valuable source of information. Multiple measurements taken over time on the same units are called *longitudinal data* or *panel data*. Longitudinal data often yield a more credible comparison of the treatment and control groups than *cross-section data* because the former contain additional information about changes over time. In the minimum-wage study, the researchers had collected the employment and wage information from the same set of restaurants before the minimum wage was increased in NJ. This pretreatment information allows several alternative designs for estimating causal effects in observational studies.

The first possibility is comparison between pre- and posttreatment measurements, which is called the *before-and-after design*. Instead of comparing the fast-food restaurants in NJ with those in PA after the increase in the NJ minimum wage, this design compares the same set of fast-food restaurants in NJ before and after the minimum wage was raised. We compute the estimate under this design as follows.

```
## full-time employment proportion in the previous period for NJ
minwageNJ$fullPropBefore <- minwageNJ$fullBefore /
    (minwageNJ$fullBefore + minwageNJ$partBefore)
## mean difference between before and after the minimum wage increase
NJdiff <- mean(minwageNJ$fullPropAfter) - mean(minwageNJ$fullPropBefore)
NJdiff

## [1] 0.02387474
```

The before-and-after analysis gives an estimate that is similar to those obtained earlier. The advantage of this design is that any confounding factor that is specific to each state is held constant because the comparison is done within NJ. The disadvantage of the before-and-after design, however, is that time-varying confounding factors can bias the resulting inference. For example, suppose that there is an upwards *time trend* in the local economy and wages and employment are improving. If this trend is not caused by the minimum-wage increase, then we may incorrectly attribute the outcome difference between the two time periods to the raise in the minimum

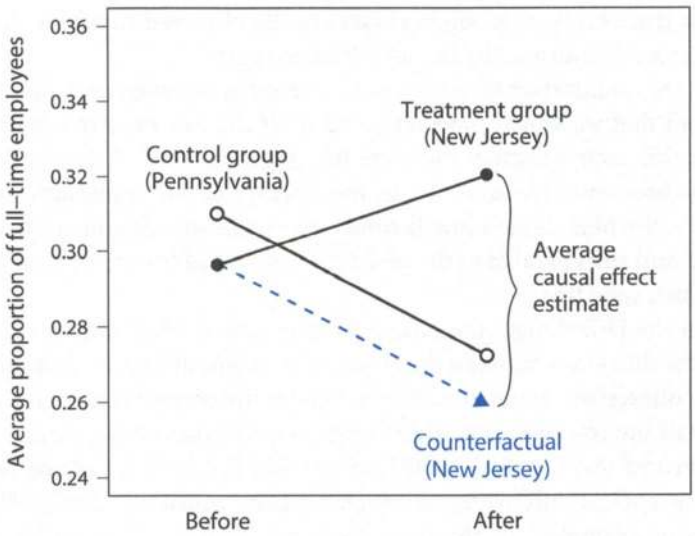


Figure 2.2. The Difference-in-Differences Design in the Minimum-Wage Study. The observed outcomes, i.e., the average proportion of full-time employees, are shown before and after the increase in the minimum wage for both the treatment group (fast-food restaurants in New Jersey; solid black circles) and the control group (restaurants in Pennsylvania; open black circles). Under the difference-in-differences design, the counterfactual outcome for the treatment group (solid blue triangle) is estimated by assuming that the time trend for the treatment group is parallel to the observed trend for the control group. The estimated average causal effect for New Jersey restaurants is indicated by the curly brace.

wage. The before-and-after design critically rests upon the nonexistence of such time trends.

The **before-and-after design** examines how the outcome variable changed from the pretreatment period to the posttreatment period for the same set of units. The design is able to adjust for any confounding factor that is specific to each unit but does not change over time. However, the design does not address possible bias due to time-varying confounders.

The *difference-in-differences* (DiD) design extends the before-and-after design to address the confounding bias due to time trends. The key assumption behind the DiD design is that the outcome variable follows a parallel trend in the absence of treatment. Figure 2.2 graphically illustrates this assumption using the minimum-wage study data. The figure shows the outcome of interest, i.e., the average proportion of full-time employees, before and after the increase in the minimum wage for both the treatment group (fast-food restaurants in NJ, indicated by the solid black circles) and the control group (restaurants in PA, represented by the open black circles). In this setting, we can estimate the counterfactual outcome for the treatment group by assuming that the time

trend for the treatment group is parallel to the observed trend for the control group. This estimate is indicated by the solid blue triangle.

Here, the counterfactual outcome of interest is the average proportion of full-time employees that we would have observed if NJ did not raise the minimum wage. We estimate this counterfactual outcome by supposing that NJ would have experienced the same economic trend as PA in the absence of the minimum-wage increase. In the figure, the blue dashed line is drawn to obtain the estimate of this counterfactual outcome and runs parallel to the observed time trend for the control group (indicated by the black solid line).

Under the DiD design, the sample average causal effect estimate for the NJ restaurants is the difference between the observed outcome after the minimum-wage increase and the counterfactual outcome derived under the parallel time trend assumption. The quantity of interest under the DiD design is called the *sample average treatment effect for the treated* (SATT). SATT differs from SATE, which is defined in equation (2.1), because it applies only to the treatment group, which consists of NJ restaurants in the current example.[4] In the figure, this estimate is indicated by the curly brace. To compute this estimate, we first calculate the difference in the outcome for the restaurants in PA after and before the minimum wage was raised in NJ. We then subtract this difference from the estimate obtained under the before-and-after design, which equals the difference in NJ after and before the minimum-wage increase. The average causal effect estimate is, therefore, given by the difference in the before-and-after differences between the treatment and control groups.

In this way, the DiD design uses the pretreatment and posttreatment measurements obtained for both the treatment and control groups. In contrast, the cross-section comparison requires only the posttreatment measurements from the two groups, and the before-and-after design utilizes the pretreatment and posttreatment measurements for the treatment group alone.

The **difference-in-differences** (DiD) design uses the following estimate of the sample average treatment effect for the treated (SATT):

$$\text{DiD estimate} = \underbrace{\left(\overline{Y}^{\text{after}}_{\text{treated}} - \overline{Y}^{\text{before}}_{\text{treated}}\right)}_{\text{difference for the treatment group}} - \underbrace{\left(\overline{Y}^{\text{after}}_{\text{control}} - \overline{Y}^{\text{before}}_{\text{control}}\right)}_{\text{difference for the control group}}.$$

The assumption is that the counterfactual outcome for the treatment group has a time trend parallel to that of the control group.

In the case of the minimum-wage study, we can compute the DiD estimate as follows.

[4] Formally, the sample average treatment effect for the treated (SATT) is the sample average of individual-level causal effect among the treated units, $\text{SATT} = \frac{1}{n_1}\sum_{i=1}^{n} T_i\{Y_i(1) - Y_i(0)\}$, where T_i is the binary treatment indicator variable and $n_1 = \sum_{i=1}^{n} T_i$ is the size of the treatment group.

```
## full-time employment proportion in the previous period for PA
minwagePA$fullPropBefore <- minwagePA$fullBefore /
    (minwagePA$fullBefore + minwagePA$partBefore)
## mean difference between before and after for PA
PAdiff <- mean(minwagePA$fullPropAfter) - mean(minwagePA$fullPropBefore)
## difference-in-differences
NJdiff - PAdiff
## [1] 0.06155831
```

The result is inconsistent with the prediction of some economists that raising the minimum wage has a negative impact on employment. To the contrary, our DiD analysis suggests that, if anything, the increase in the minimum wage may have led to a small rise in the proportion of full-time employees in NJ fast-food restaurants. The DiD estimate is greater than the before-and-after estimate, which reflected a negative trend in PA.

When does the DiD design fail? The DiD design yields an invalid estimate of causal effect if the time trend of the counterfactual outcome for the treatment group is not parallel to the observed time trend for the control group. We cannot verify this assumption because the counterfactual time trend for the treatment group is unobserved. However, in some cases, we can increase the credibility of this assumption. For example, if researchers had collected employment information from the restaurants in earlier time periods, then they could have examined whether the proportion of full-time employees in NJ restaurants had changed parallel to that of PA restaurants when the minimum wage had not been raised.

2.6 Descriptive Statistics for a Single Variable

So far, we have been examining the average outcome as the quantity of interest, but it is also possible to consider some other statistics of outcome. As the final topic of this chapter, we discuss how to numerically summarize the distribution of a single variable using *descriptive statistics*. We have already seen some examples of descriptive statistics, including the range (i.e., minimum and maximum values), median, and mean. In this section, we introduce other commonly used univariate statistics to describe the distribution of a single variable.

2.6.1 QUANTILES

We begin by introducing *quantiles*, which divide a set of observations into groups based on the magnitude of the variable. An example of quantiles is the *median*, which divides the data into two groups, one with lower data values and the other with higher values. That is, the median of a variable equals the middle value if the total number of observations is odd, whereas the median is the average of two middle values if the total number of observations is even (because there is no single middle value in this case). For example, the median of {1, 3, 4, 10} is 3.5, which is the average of the middle values 3 and 4, because this example has an even number of values. Meanwhile, the mean of this vector is 4.5.

While both the mean and median measure the center of the distribution, the mean is more sensitive to *outliers*. For example, a single observation of extreme value can dramatically change the mean but it will not affect the median as much. The median of $\{1, 3, 4, 10, 82\}$ is 4, but the mean now increases to 20. In the minimum-wage data, the mean and median wages are similar. For example, the median wage before the minimum-wage increase is \$4.50, which is close to its mean of \$4.62.

The **median** of a variable x is defined as:

$$\text{median} = \begin{cases} x_{((n+1)/2)} & \text{if } n \text{ is odd,} \\ \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right) & \text{if } n \text{ is even,} \end{cases} \tag{2.2}$$

where $x_{(i)}$ denotes the value of the ith smallest observation for variable x and n is the sample size. The median is less sensitive to outliers than the **mean** and hence is a more robust measure of the center of a distribution.

To examine the robustness of previous findings, we examine how the increase in the minimum wage influenced the proportion of full-time employees in terms of the median rather than the mean. The median of a variable can be computed by using the `median()` function.

```
## cross-section comparison between NJ and PA
median(minwageNJ$fullPropAfter) - median(minwagePA$fullPropAfter)
## [1] 0.07291667

## before and after comparison
NJdiff.med <- median(minwageNJ$fullPropAfter) -
    median(minwageNJ$fullPropBefore)
NJdiff.med
## [1] 0.025

## median difference-in-differences
PAdiff.med <- median(minwagePA$fullPropAfter) -
    median(minwagePA$fullPropBefore)
NJdiff.med - PAdiff.med
## [1] 0.03701923
```

These results are largely consistent with those of the previous analysis, though the DiD estimate is smaller than before. Again, there is little evidence for the hypothesis that increasing the minimum wage decreases full-time employment. If anything, it may have instead slightly increased full-time employment.

To obtain a more complete description of the distribution, we can use *quartiles*, which divide the data into four groups. The *first quartile* (or *lower quartile*) is the

value under which 25% of the observations fall, while the proportion of observations below the *third quartile* (or *upper quartile*) is 75%. The *second quartile* is equal to the median. The quartiles are a part of the output from the `summary()` function along with the minimum, mean, and maximum values. In addition, the difference between the upper and lower quartiles (i.e., 75th percentile and the 25th percentile) is called the *interquartile range* or *IQR*. That is, the IQR represents the range that contains 50% of the data, thereby measuring the spread of a distribution. This statistic can be computed by the `IQR()` function.

```
## summary shows quartiles as well as minimum, maximum, and mean
summary(minwageNJ$wageBefore)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.25    4.25    4.50    4.61    4.87    5.75

summary(minwageNJ$wageAfter)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.000   5.050   5.050   5.081   5.050   5.750

## interquartile range
IQR(minwageNJ$wageBefore)
## [1] 0.62

IQR(minwageNJ$wageAfter)
## [1] 0
```

This analysis shows that before the minimum-wage increase, the distribution of wages ranged from \$4.25 to \$5.75 with 75% of the fast-food restaurants in NJ having wages of \$4.87 per hour or less. However, after the minimum wage was raised to \$5.05, many restaurants raised their wages just to the new minimum wage but not any higher. As a result, both the lower and upper quartiles are equal to \$5.05, reducing the IQR from \$0.62 to \$0.

Finally, quartiles belong to a class of general statistics called *quantiles*, which divide the observations into a certain number of equally sized groups. Other quantiles include *terciles* (which divide the data into 3 groups), *quintiles* (5 groups), *deciles* (10 groups), and *percentiles* (100 groups). The `quantile()` function can generate any quantiles by specifying the `probs` argument. This argument takes a sequence of probabilities, indicating how the data should be divided up. For example, the deciles of the wage variable are obtained using the `seq()` function to create a sequence of numbers `0, 0.1, ..., 0.9, 1`.

```
## deciles (10 groups)
quantile(minwageNJ$wageBefore, probs = seq(from = 0, to = 1, by = 0.1))
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
## 4.25 4.25 4.25 4.25 4.50 4.50 4.65 4.75 5.00 5.00 5.75
```

```
quantile(minwageNJ$wageAfter, probs = seq(from = 0, to = 1, by = 0.1))

##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
## 5.00 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.15 5.75
```

We find that at least 90% of the fast-food restaurants in NJ set their wages to \$5.05 or higher after the law was enacted. In contrast, before the increase in the minimum wage, there were few restaurants that offered wages of \$5.05 or higher. Thus, the law had a dramatic effect on raising the wage to the new minimum wage, but no higher than that. In fact, the highest wage stayed unchanged at \$5.75 even after the minimum wage was increased.

Quantiles represent a set of data values that divide observations into a certain number of equally sized groups. They include quartiles (dividing the observations into 4 groups) and percentiles (100 groups):

- 25th percentile = lower quartile;
- 50th percentile = median;
- 75th percentile = upper quartile.

The difference between the upper and lower quartiles is called the **interquartile range** and measures the spread of a distribution.

2.6.2 STANDARD DEVIATION

We have used the range and quantiles (including the IQR) to describe the spread of a distribution. Another commonly used measure is *standard deviation*. Before introducing standard deviation, we first describe a statistic called the *root mean square* or *RMS*. The RMS describes the magnitude of a variable and is defined as

$$\begin{aligned} \text{RMS} &= \sqrt{\text{mean of squared entries}} \\ &= \sqrt{\frac{\text{entry1}^2 + \text{entry2}^2 + \cdots}{\text{number of entries}}} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}. \end{aligned} \tag{2.3}$$

Equation (2.3) gives the formal mathematical definition. The equation exactly follows its name—square each entry, compute the mean, and then take the square root.

While the mean describes the center of the distribution, the RMS represents the average absolute magnitude of each data entry, ignoring the sign of the entry (e.g., the absolute magnitude or *absolute value* of -2 is 2 and is written as $|-2|$). For example, the mean of $\{-2, -1, 0, 1, 2\}$ is 0 but its RMS is $\sqrt{2}$. In the minimum-wage data, we can compute the RMS of the change in the proportion of full-time employees before and after the increase in the minimum wage, which is quite different from its mean.

```
sqrt(mean((minwageNJ$fullPropAfter - minwageNJ$fullPropBefore)^2))
## [1] 0.3014669

mean(minwageNJ$fullPropAfter - minwageNJ$fullPropBefore)
## [1] 0.02387474
```

Thus, on average, the absolute magnitude of change in the proportion of full-time employees, after the minimum wage was raised, is about 0.3. This represents a relatively large change even though the average difference is close to zero.

Using the RMS, we can define the sample *standard deviation* as the average deviation of each data entry from its mean. Therefore, the standard deviation measures the spread of a distribution by quantifying how far away data points are, on average, from their mean. Specifically, the standard deviation is defined as the RMS of deviation from the average:

$$\begin{aligned} \text{standard deviation} &= \text{RMS of deviation from average} \\ &= \sqrt{\frac{(\text{entry1} - \text{mean})^2 + (\text{entry2} - \text{mean})^2 + \cdots}{\text{number of entries}}} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}. \end{aligned} \tag{2.4}$$

In some cases, one uses $n - 1$ instead of n in the denominator of equation (2.4) for a reason that will become clear in chapter 7, but this results in only a minor difference so long as one has enough data. We note that few data points are more than 2 or 3 standard deviations away from the mean. Hence, knowing the standard deviation helps researchers understand the approximate range of the data as well. Finally, the square of the standard deviation is called the *variance* and represents the average squared deviation from the mean. We will study variance more closely in later chapters. Variance is more difficult to interpret than standard deviation, but it has useful analytical properties, as shown in chapter 6.

The sample **standard deviation** measures the average deviation from the mean and is defined as

$$\text{standard deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \text{or} \quad \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

where $\bar{x}$ represents the sample mean, i.e., $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and n is the sample size. Few data points lie outside 2 or 3 standard deviations away from the mean. The square of the standard deviation is called the **variance**.

In R, we can easily compute the standard deviation using the `sd()` function (this function uses $n - 1$ in its denominator). The `var()` function returns the sample variance. The examples from the minimum-wage data are given here.

```
## standard deviation
sd(minwageNJ$fullPropBefore)
## [1] 0.2304592

sd(minwageNJ$fullPropAfter)
## [1] 0.2510016

## variance
var(minwageNJ$fullPropBefore)
## [1] 0.05311145

var(minwageNJ$fullPropAfter)
## [1] 0.0630018
```

The results indicate that, on average, the proportion of full-time employees for a NJ fast-food restaurant is approximately 0.2 away from its mean. We find that for this variable the standard deviation did not change much after the minimum wage had been increased.

2.7 Summary

We began this chapter with the analysis of an experimental study concerning racial discrimination in the labor market. The **fundamental problem of causal inference** is the fact that we observe only one of two potential outcomes and yet the estimation of causal effect involves comparison between counterfactual and factual outcomes. This chapter also introduced various research design strategies to infer counterfactual outcomes from observed data. It is important to understand the assumptions that underlie each research design as well as their strengths and weaknesses.

In **randomized controlled experiments (RCTs)**, a simple comparison of the treatment and control groups enables researchers to estimate the causal effects of treatment. By randomizing the treatment assignment, we can ensure that the treatment and control groups are, on average, identical to each other in all observed and unobserved characteristics except for the receipt of treatment. Consequently, any average difference between the treatment and control groups can be attributed to the treatment. While RCTs tend to yield internally valid estimates of causal effects, they often suffer from a lack of external validity, which makes it difficult to generalize empirical conclusions to a relevant population in real-world settings.

In **observational studies**, researchers do not directly conduct interventions. Since some subjects may self-select into the treatment group, the difference in outcome between the treatment and control groups can be attributed to factors other than the receipt of treatment. Thus, while observational studies often have stronger external validity, this advantage typically comes with compromises in internal validity. When the treatment assignment is not randomized, we must confront the possibility of confounding bias in observational studies using statistical control. The existence of confounders that are associated with both the treatment and outcome means that a simple comparison of the two groups yields misleading inference. We introduced various research design strategies to reduce such bias, including subclassification, before-and-after design, and difference-in-differences design.

Finally, we learned how to subset data in various ways using R. Subsetting can be done using logical values, relational operators, and conditional statements. We also introduced a number of descriptive statistics that are useful for summarizing each variable in a data set. They include the mean, median, quantiles, and standard deviation. R provides a set of functions that enable researchers to compute these and other descriptive statistics from their data sets.

2.8 Exercises

2.8.1 EFFICACY OF SMALL CLASS SIZE IN EARLY EDUCATION

The STAR (Student–Teacher Achievement Ratio) Project is a four-year *longitudinal study* examining the effect of class size in early grade levels on educational performance and personal development.[5] A longitudinal study is one in which the same participants are followed over time. This particular study lasted from 1985 to 1989 and involved 11,601 students. During the four years of the study, students were randomly assigned to small classes, regular-sized classes, or regular-sized classes with an aid. In all, the experiment cost around $12 million. Even though the program stopped in 1989 after the first kindergarten class in the program finished third grade, the collection of various measurements (e.g., performance on tests in eighth grade, overall high-school GPA) continued through to the end of participants' high-school attendance.

We will analyze just a portion of this data to investigate whether the small class sizes improved educational performance or not. The data file name is `STAR.csv`, which is

[5] This exercise is in part based on Frederick Mosteller (1995) "The Tennessee study of class size in the early school grades." *The Future of Children*, vol. 5, no. 2, pp. 113–127.

Table 2.6. STAR Project Data.

Variable	*Description*
`race`	student's race (white = 1, black = 2, Asian = 3, Hispanic = 4, Native American = 5, others = 6)
`classtype`	type of kindergarten class (small = 1, regular = 2, regular with aid = 3)
`g4math`	total scaled score for the math portion of the fourth-grade standardized test
`g4reading`	total scaled score for the reading portion of the fourth-grade standardized test
`yearssmall`	number of years in small classes
`hsgrad`	high-school graduation (did graduate = 1, did not graduate = 0)

in CSV format. The names and descriptions of variables in this data set are displayed in table 2.6. Note that there are a fair amount of missing values in this data set, which arise, for example, because some students left a STAR school before third grade, or did not enter a STAR school until first grade.

1. Create a new factor variable called `kinder` in the data frame. This variable should recode `classtype` by changing integer values to their corresponding informative labels (e.g., change `1` to `small` etc.). Similarly, recode the `race` variable into a factor variable with four levels (`white`, `black`, `hispanic`, `others`) by combining the Asian and Native American categories with the others category. For the `race` variable, overwrite the original variable in the data frame rather than creating a new one. Recall that `na.rm = TRUE` can be added to functions in order to remove missing data (see section 1.3.5).

2. How does performance on fourth-grade reading and math tests for those students assigned to a small class in kindergarten compare with those assigned to a regular-sized class? Do students in the smaller classes perform better? Use means to make this comparison while removing missing values. Give a brief substantive interpretation of the results. To understand the size of the estimated effects, compare them with the standard deviation of the test scores.

3. Instead of just comparing average scores of reading and math tests between those students assigned to small classes and those assigned to regular-sized classes, look at the entire range of possible scores. To do so, compare a high score, defined as the 66th percentile, and a low score (the 33rd percentile) for small classes with the corresponding score for regular classes. These are examples of *quantile treatment effects*. Does this analysis add anything to the analysis based on mean in the previous question?

4. Some students were in small classes for all four years that the STAR program ran. Others were assigned to small classes for only one year and had either regular-sized classes or regular-sized classes with an aid for the rest. How many students

Table 2.7. Gay Marriage Data.

Variable	*Description*
`study`	source of the data (`1` = study 1, `2` = study 2)
`treatment`	five possible treatment assignment options
`wave`	survey wave (a total of seven waves)
`ssm`	five-point scale on same-sex marriage, higher scores indicate support.

of each type are in the data set? Create a contingency table of proportions using the `kinder` and `yearssmall` variables. Does participation in more years of small classes make a greater difference in test scores? Compare the average and median reading and math test scores across students who spent different numbers of years in small classes.

5. Examine whether the STAR program reduced achievement gaps across different racial groups. Begin by comparing the average reading and math test scores between white and minority students (i.e., blacks and Hispanics) among those students who were assigned to regular-sized classes with no aid. Conduct the same comparison among those students who were assigned to small classes. Give a brief substantive interpretation of the results of your analysis.

6. Consider the long-term effects of kindergarten class size. Compare high-school graduation rates across students assigned to different class types. Also, examine whether graduation rates differ depending on the number of years spent in small classes. Finally, as in the previous question, investigate whether the STAR program has reduced the racial gap between white and minority students' graduation rates. Briefly discuss the results.

2.8.2 CHANGING MINDS ON GAY MARRIAGE

In this exercise, we analyze the data from two experiments in which households were canvassed for support on gay marriage.[6] Note that the original study was later retracted due to allegations of fabricated data; we will revisit this issue in a follow-up exercise (see section 3.9.1). In this exercise, however, we analyze the original data while ignoring the allegations.

Canvassers were given a script leading to conversations that averaged about twenty minutes. A distinctive feature of this study is that gay and straight canvassers were randomly assigned to households, and canvassers revealed whether they were straight or gay in the course of the conversation. The experiment aims to test the "contact hypothesis," which contends that out-group hostility (towards gay people in this case) diminishes when people from different groups interact with one another. The data file is `gay.csv`, which is a CSV file. Table 2.7 presents the names and descriptions

[6] This exercise is based on the following article: Michael J. LaCour and Donald P. Green (2015) "When contact changes minds: An experiment on transmission of support for gay equality." *Science*, vol. 346, no. 6215, pp. 1366–1369.

of the variables in this data set. Each observation of this data set is a respondent giving a response to a four-point survey item on same-sex marriage. There are two different studies in this data set, involving interviews during seven different time periods (i.e., seven waves). In both studies, the first wave consists of the interview before the canvassing treatment occurs.

1. Using the baseline interview wave before the treatment is administered, examine whether randomization was properly conducted. Base your analysis on the three groups of study 1: "same-sex marriage script by gay canvasser," "same-sex marriage script by straight canvasser" and "no contact." Briefly comment on the results.

2. The second wave of the survey was implemented two months after canvassing. Using study 1, estimate the average treatment effects of gay and straight canvassers on support for same-sex marriage, separately. Give a brief interpretation of the results.

3. The study contained another treatment that involves contact, but does not involve using the gay marriage script. Specifically, the authors used a script to encourage people to recycle. What is the purpose of this treatment? Using study 1 and wave 2, compare outcomes from the treatment "same-sex marriage script by gay canvasser" to "recycling script by gay canvasser." Repeat the same for straight canvassers, comparing the treatment "same-sex marriage script by straight canvasser" to "recycling script by straight canvasser." What do these comparisons reveal? Give a substantive interpretation of the results.

4. In study 1, the authors reinterviewed the respondents six different times (in waves 2 to 7) after treatment, at two-month intervals. The last interview, in wave 7, occurs one year after treatment. Do the effects of canvassing last? If so, under what conditions? Answer these questions by separately computing the average effects of straight and gay canvassers with the same-sex marriage script for each of the subsequent waves (relative to the control condition).

5. The researchers conducted a second study to replicate the core results of the first study. In this study, same-sex marriage scripts are given only by gay canvassers. For study 2, use the treatments "same-sex marriage script by gay canvasser" and "no contact" to examine whether randomization was appropriately conducted. Use the baseline support from wave 1 for this analysis.

6. For study 2, estimate the treatment effects of gay canvassing using data from wave 2. Are the results consistent with those of study 1?

7. Using study 2, estimate the average effect of gay canvassing at each subsequent wave and observe how it changes over time. Note that study 2 did not have a fifth or sixth wave, but the seventh wave occurred one year after treatment, as in study 1. Draw an overall conclusion from both study 1 and study 2.

Table 2.8. Leader Assassination Data.

Variable	*Description*
`country`	country
`year`	year
`leadername`	name of the leader who was targeted
`age`	age of the targeted leader
`politybefore`	average polity score of the country during the three-year period prior to the attempt
`polityafter`	average polity score of the country during the three-year period after the attempt
`civilwarbefore`	1 if the country was in civil war during the three-year period prior to the attempt, 0 otherwise
`civilwarafter`	1 if the country was in civil war during the three-year period after the attempt, 0 otherwise
`interwarbefore`	1 if the country was in international war during the three-year period prior to the attempt, 0 otherwise
`interwarafter`	1 if the country was in international war during the three-year period after the attempt, 0 otherwise
`result`	result of the assassination attempt

2.8.3 SUCCESS OF LEADER ASSASSINATION AS A NATURAL EXPERIMENT

One longstanding debate in the study of international relations concerns the question of whether individual political leaders can make a difference. Some emphasize that leaders with different ideologies and personalities can significantly affect the course of a nation. Others argue that political leaders are severely constrained by historical and institutional forces. Did individuals like Hitler, Mao, Roosevelt, and Churchill make a big difference? The difficulty of empirically testing these arguments stems from the fact that the change of leadership is not random and there are many confounding factors to be adjusted for.

In this exercise, we consider a *natural experiment* in which the success or failure of assassination attempts is assumed to be essentially random.[7] Each observation of the CSV data set `leaders.csv` contains information about an assassination attempt. Table 2.8 presents the names and descriptions of variables in this leader assassination data set. The `polity` variable represents the so-called *polity score* from the Polity Project. The Polity Project systematically documents and quantifies the regime types of all countries in the world from 1800. The polity score is a 21-point scale ranging from −10 (hereditary monarchy) to 10 (consolidated democracy). The `result` variable is a 10-category factor variable describing the result of each assassination attempt.

1. How many assassination attempts are recorded in the data? How many countries experience at least one leader assassination attempt? (The `unique()` function,

[7] This exercise is based on the following article: Benjamin F. Jones and Benjamin A. Olken (2009) "Hit or miss? The effect of assassinations on institutions and war." *American Economic Journal: Macroeconomics*, vol. 1, no. 2, pp. 55–87.

which returns a set of unique values from the input vector, may be useful here.) What is the average number of such attempts (per year) among these countries?

2. Create a new binary variable named `success` that is equal to `1` if a leader dies from the attack and `0` if the leader survives. Store this new variable as part of the original data frame. What is the overall success rate of leader assassination? Does the result speak to the validity of the assumption that the success of assassination attempts is randomly determined?

3. Investigate whether the average polity score over three years prior to an assassination attempt differs on average between successful and failed attempts. Also, examine whether there is any difference in the age of targeted leaders between successful and failed attempts. Briefly interpret the results in light of the validity of the aforementioned assumption.

4. Repeat the same analysis as in the previous question, but this time using the country's experience of civil and international war. Create a new binary variable in the data frame called `warbefore`. Code the variable such that it is equal to `1` if a country is in either civil or international war during the three years prior to an assassination attempt. Provide a brief interpretation of the result.

5. Does successful leader assassination cause democratization? Does successful leader assassination lead countries to war? When analyzing these data, be sure to state your assumptions and provide a brief interpretation of the results.

Chapter 3

Measurement

Not everything that can be counted counts, and not everything that counts can be counted.
—William Bruce Cameron, *Informal Sociology*

Measurement plays a central role in social science research. In this chapter, we first discuss survey methodology, which is perhaps the most common mode of data collection. For example, the minimum-wage study discussed in chapter 2 used a survey to measure information about employment at each fast-food restaurant. Surveys are also effective tools for making inferences about a large target population of interest from a relatively small sample of randomly selected units. In addition to surveys, we also discuss the use of latent concepts, such as ideology, that are essential for social science research. These concepts are fundamentally unobservable and must be measured using a theoretical model. Thus, issues of measurement often occupy the intersection of theoretical and empirical analyses in the study of human behavior. Finally, we introduce a basic clustering method, which enables researchers to conduct an exploratory analysis of data by discovering interesting patterns. We also learn how to plot data in various ways and compute relevant descriptive statistics in R.

3.1 Measuring Civilian Victimization during Wartime

After the September 11 attacks, the United States and its allies invaded Afghanistan with the goal of dismantling al-Qaeda, which had been operating there under the protection of the Taliban government. In 2003, the North Atlantic Treaty Organization (NATO) became involved in the conflict, sending in a coalition of international troops organized under the name of the International Security Assistance Force (ISAF). To wage this war against the Taliban insurgency, the ISAF engaged in a "hearts and minds" campaign, combining economic assistance, service delivery, and protection in order to win the support of civilians. To evaluate the success of such a campaign, it is essential to measure and understand civilians' experiences and sentiments during the war. However, measuring the experiences and opinions of civilians during wartime is a challenging task because of harsh security conditions, posing potential threats to

Table 3.1. Afghanistan Survey Data.

Variable	*Description*
`province`	province where the respondent lives
`district`	district where the respondent lives
`village.id`	ID of the village where the respondent lives
`age`	age of the respondent
`educ.years`	years of education of the respondent
`employed`	whether the respondent is employed
`income`	monthly income of the respondent (five levels)
`violent.exp.ISAF`	whether the respondent experienced violence by ISAF
`violent.exp.taliban`	whether the respondent experienced violence by the Taliban
`list.group`	randomly assigned group for the list experiment (`control`, `ISAF`, `taliban`)
`list.response`	response to the list experiment question (`0`–`4`)

interviewers and respondents. This means that respondents may inaccurately answer survey questions in order to avoid giving socially undesirable responses.

A group of social scientists conducted a public opinion *survey* in southern Afghanistan, the heartland of the insurgency.[1] The survey was administered to a sample of 2754 respondents between January and February 2011. The researchers note that the participation rate was 89%. That is, they originally contacted 3097 males and 343 of them refused to take the survey. Because local culture prohibited interviewers from talking to female citizens, the respondents were all males.

We begin by summarizing the characteristics of respondents in terms of age, years of education, employment, and monthly income in Afghani (the local currency). The CSV file `afghan.csv` contains the survey data and can be loaded via the `read.csv()` function. The names and descriptions of the variables are given in table 3.1. We use the `summary()` function to provide numerical summaries of several variables.

```
## load data
afghan <- read.csv("afghan.csv")
## summarize variables of interest
summary(afghan$age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   22.00   30.00   32.39   40.00   80.00
```

[1] This section is based on the following two articles: Jason Lyall, Graeme Blair, and Kosuke Imai (2013) "Explaining support for combatants during wartime: A survey experiment in Afghanistan." *American Political Science Review*, vol. 107, no. 4 (November), pp. 679–705 and Graeme Blair, Kosuke Imai, and Jason Lyall (2014) "Comparing and combining list and endorsement experiments: Evidence from Afghanistan." *American Journal of Political Science*, vol. 58, no. 4 (October), pp. 1043–1063.

```
summary(afghan$educ.years)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   4.002   8.000  18.000

summary(afghan$employed)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5828  1.0000  1.0000

summary(afghan$income)

##   10,001-20,000    2,001-10,000   20,001-30,000
##             616            1420              93
## less than 2,000     over 30,000            NA's
##             457              14             154
```

We observe that the average age of the respondents is 32, a large fraction of them have very little education, and approximately 60% of the respondents are employed. Most respondents have a monthly income of less than 10,000 Afghani, which is about 200 dollars.

While civilians are often victimized during war, it is difficult to systematically measure the extent to which attacks against civilians occur. A survey measure, though it is based on self-reporting, is one possible way to quantify civilian victimization. In this survey, the interviewers asked the following question: "Over the past year, have you or anyone in your family suffered harm due to the actions of the Foreign Forces / the Taliban?" They explained to the respondents that the phrase "harm" refers to physical injury, as well as property damage. We analyze the `violent.exp.ISAF` and `violent.exp.taliban` variables, which represent whether the respondents were harmed by the ISAF and the Taliban, respectively.

```
prop.table(table(ISAF = afghan$violent.exp.ISAF,
                 Taliban = afghan$violent.exp.taliban))

##      Taliban
## ISAF         0         1
##    0 0.4953445 0.1318436
##    1 0.1769088 0.1959032
```

Using the `table()` and `prop.table()` functions, which were introduced in chapter 2, the analysis shows that over the past year, 37%(= 17.7% + 19.6%) and 33% (= 13.2% + 19.6%) of the respondents were victimized by the ISAF (second row) and the Taliban (second column), respectively. Approximately 20% of the respondents suffered from physical or property damage caused by both parties. This finding suggests that Afghan civilians were victimized (or at least they perceived that they were being victimized) by both the ISAF and the Taliban to a similar extent, rather than one warring party disproportionately harming civilians.

3.2 Handling Missing Data in R

In many surveys, researchers may encounter nonresponse because either respondents refuse to answer some questions or they simply do not know the answer. Such missing values are also common in other types of data. For example, many developing countries lack certain official statistics such as the gross domestic product (GDP) or unemployment rate. In R, missing data are coded as NA. For example, in the Afghanistan survey, we saw in the above analysis that 154 respondents did not provide their income. Since NA is a special value reserved for missing data, we can count the number of missing observations using the `is.na()` function. This function returns a logical value of TRUE if its argument is NA and yields FALSE otherwise.

```
## print income data for first 10 respondents
head(afghan$income, n = 10)
##  [1] 2,001-10,000  2,001-10,000  2,001-10,000  2,001-10,000
##  [5] 2,001-10,000  <NA>          10,001-20,000 2,001-10,000
##  [9] 2,001-10,000  <NA>
## 5 Levels: 10,001-20,000 2,001-10,000 ... over 30,000

## indicate whether respondents' income is missing
head(is.na(afghan$income), n = 10)
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [10]  TRUE
```

Here, we see that the sixth and tenth respondents are not reporting their monthly income and hence are coded as NA. The syntax `is.na(afghan$income)` returns a vector of logical values, each indicating whether the corresponding respondent provided an answer to the income question. Thus, the sixth and tenth elements of the output from this syntax are TRUE. Given this function, it is now straightforward to count the total number and proportion of missing data for this variable.

```
sum(is.na(afghan$income)) # count of missing values
## [1] 154

mean(is.na(afghan$income)) # proportion missing
## [1] 0.05591866
```

Some R functions treat missing data differently from other data. For example, the `mean()` function returns NA when a variable contains at least one missing value. Fortunately, the `mean()` function takes an additional argument `na.rm`, which can be set to TRUE so that missing data are removed before the function is applied. Many other functions, including `max()`, `min()`, and `median()`, take this argument as well.

```
x <- c(1, 2, 3, NA)
mean(x)
## [1] NA

mean(x, na.rm = TRUE)
## [1] 2
```

In our data, the application of the `table()` function above ignored missing data, as if observations with missing values were not part of the data set. We can tell these functions to explicitly account for missing data. This can be done by setting the additional argument `exclude` to `NULL` so that no data including a missing value is excluded.

```
prop.table(table(ISAF = afghan$violent.exp.ISAF,
                 Taliban = afghan$violent.exp.taliban, exclude = NULL))
##        Taliban
## ISAF              0           1        <NA>
##    0     0.482933914 0.128540305 0.007988381
##    1     0.172476398 0.190994916 0.007988381
##    <NA>  0.002541757 0.002904866 0.003631082
```

We find that almost all respondents answered the victimization questions. Indeed, the nonresponse rates for these questions are less than 2%. The nonresponse rates for the Taliban and ISAF victimization questions can be obtained by adding the entries of the final column and those of the final row of the above generated table, respectively. It appears that the Afghan civilians are willing to answer questions about their experiences of violence.

Finally, the `na.omit()` function provides a straightforward way to remove all observations with at least one missing value from a data frame. The function then returns another data frame without these observations. However, we should note that this operation will result in *listwise deletion*, which eliminates an entire observation if at least one of its variables has a missing value. For example, if a respondent answers every question asked of him except for the question about income, listwise deletion would completely remove all of his information from the data, including the responses to the questions that he did answer. In our Afghanistan survey data, other variables that we have not yet discussed also have missing data. As a result, applying the `na.omit()` function to the `afghan` data frame returns a subset of the data with far fewer observations than applying the same function to the `income` variable alone.

```
afghan.sub <- na.omit(afghan) # listwise deletion
nrow(afghan.sub)
## [1] 2554
```

```
length(na.omit(afghan$income))
## [1] 2600
```

We find that the procedure of listwise deletion yields a data set of 2554 observations, whereas a total of 2600 respondents answered the income question. The difference represents the number of respondents who did answer the income question but refused to answer at least one other question in the survey.

3.3 Visualizing the Univariate Distribution

Up until now, we have been summarizing the distribution of each variable in a data set using descriptive statistics such as the mean, median, and quantiles. However, it is often helpful to visualize the distribution itself. In this section, we introduce several ways to visualize the distribution of a single variable in R. When making a figure in RStudio, you may occasionally encounter the error message "figure margins too large." We can solve this problem by increasing the size of the plots pane.

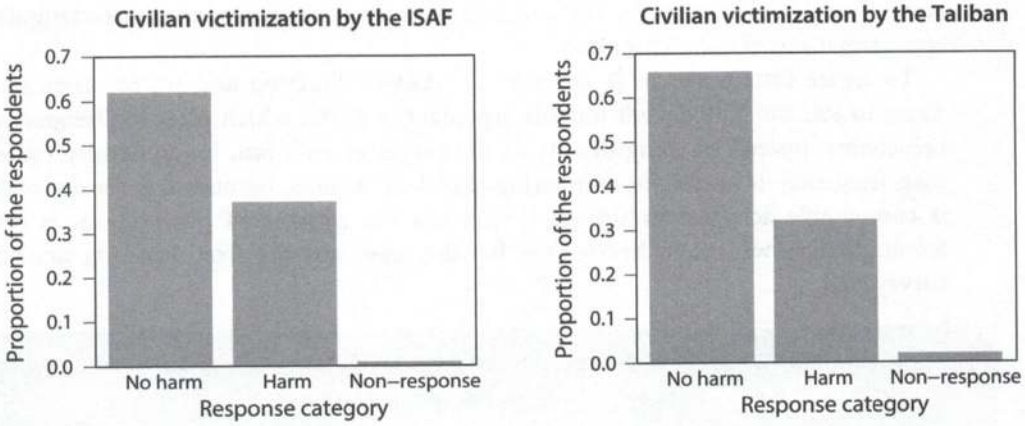
3.3.1 BAR PLOT

To summarize the distribution of a *factor variable* or *factorial variable* with several categories (see section 2.2.5), a simple table with counts or proportions, as produced above using the `table()` and `prop.table()` functions, is often sufficient. However, it is also possible to use a *bar plot* to visualize the distribution. In R, the `barplot()` function takes a vector of height and displays a bar plot in a separate graphical window. In this example, the vector of height represents the proportion of respondents in each response category.

```
## a vector of proportions to plot
ISAF.ptable <- prop.table(table(ISAF = afghan$violent.exp.ISAF,
                                exclude = NULL))
ISAF.ptable
## ISAF
##           0           1        <NA>
## 0.619462600 0.371459695 0.009077705

## make bar plots by specifying a certain range for y-axis
barplot(ISAF.ptable,
        names.arg = c("No harm", "Harm", "Nonresponse"),
        main = "Civilian victimization by the ISAF",
        xlab = "Response category",
        ylab = "Proportion of the respondents", ylim = c(0, 0.7))
## repeat the same for victimization by the Taliban
Taliban.ptable <- prop.table(table(Taliban = afghan$violent.exp.taliban,
                           exclude = NULL))
```

```
barplot(Taliban.ptable,
        names.arg = c("No harm", "Harm", "Nonresponse"),
        main = "Civilian victimization by the Taliban",
        xlab = "Response category",
        ylab = "Proportion of the respondents", ylim = c(0, 0.7))
```

The plots in this book, including these below, may appear different from those produced by running the corresponding code in R.



We immediately see that the distributions for civilian victimization by the ISAF and the Taliban are quite similar. In addition, the nonresponse rate is equally low for both variables. Note that `names.arg` is an optional argument unique to the `barplot()` function and takes a vector of characters specifying the label for each bar. The above syntax also illustrates the use of several arguments that are common to other plot functions and are summarized here:

- `main`: a character string, i.e., a series of characters in double quotes, for the main title of the plot
- `ylab`, `xlab`: character strings for labeling the vertical axis (i.e., y-axis) and the horizontal axis (i.e., x-axis), respectively (R will automatically set these arguments to the default labels if left unspecified)
- `ylim`, `xlim`: numeric vectors of length 2 specifying the interval for the y-axis and x-axis, respectively (R will automatically set these arguments if left unspecified)

3.3.2 HISTOGRAM

The *histogram* is a common method for visualizing the distribution of a *numeric variable* rather than a factor variable. Suppose that we would like to plot the histogram for the `age` variable in our Afghanistan survey data. To do this, we first discretize the variable by creating *bins* or intervals along the variable of interest. For example, we may use 5 years as the size of each bin for the `age` variable, which results in the intervals [15, 20), [20, 25), [25, 30), and so on. Recall from an exercise of chapter 1 (see section 1.5.2) that in mathematics square brackets, [and], include the limit, whereas

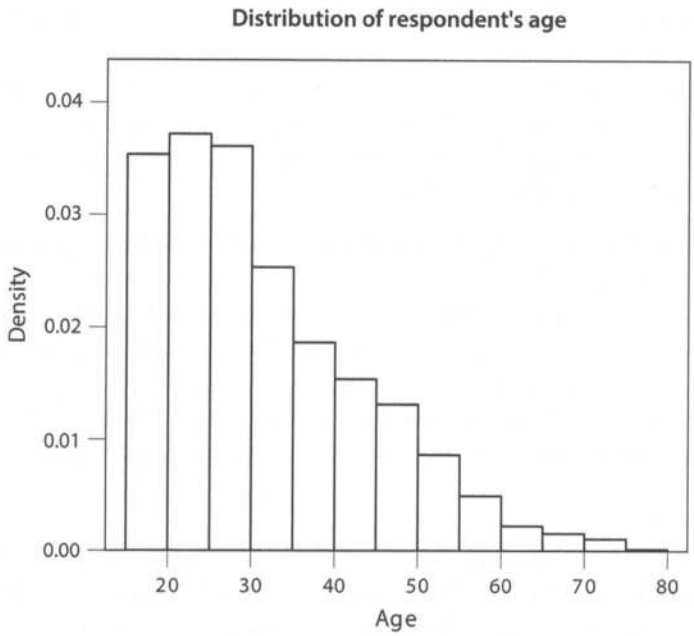
parentheses, (and), exclude it. For example, [20, 25) represents the age range that is greater than or equal to 20 years old and less than 25 years old. We then count the number of observations that fall within each bin. Finally, we compute the *density* for each bin, which is the height of the bin and is defined as

$$\text{density} = \frac{\text{proportion of observations in the bin}}{\text{width of the bin}}.$$

We often care about not the exact value of each density, but rather the variable's distribution as shown by the relationship of the different bins' densities to one another within a histogram. We can therefore think of histograms as rectangular approximations of the distribution.

To create histograms in R, we use the `hist()` function and set the argument `freq` to `FALSE`. The default for this argument is `TRUE`, which plots the frequency, i.e., counts, instead of using density as the height of each bin. Using density rather than frequency is useful for comparing two distributions, because the density scale is comparable across distributions even when the number of observations is different. Below, we create histograms for the `age` variable from the Afghanistan survey data.

```
hist(afghan$age, freq = FALSE, ylim = c(0, 0.04), xlab = "Age",
     main = "Distribution of respondent's age")
```



Importantly, the area of each bin in a histogram equals the proportion of observations that fall in that bin. Therefore, in general, we interpret the density scale, the unit of the vertical axis, as percentage per horizontal unit. In the age example, the density is measured as percentage per year. This implies that density is not a proportion and

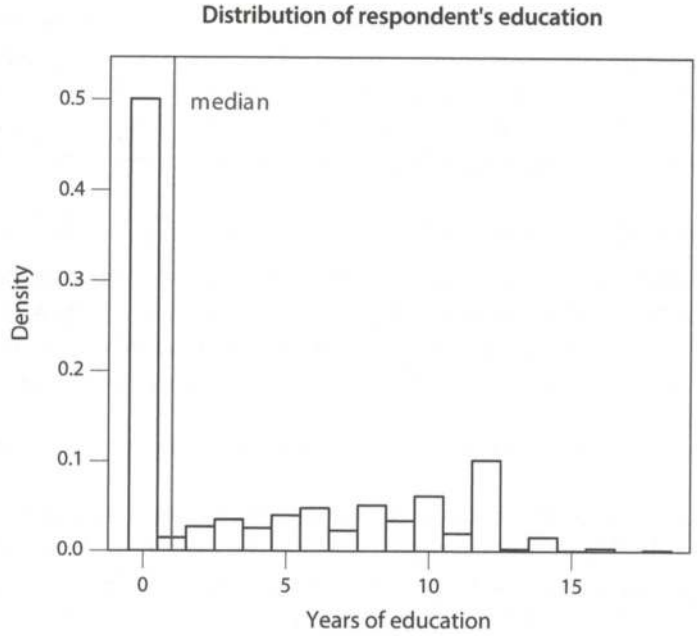
hence the height of each bin can exceed 1. On the other hand, the area of each bin represents the percentage of observations it contains, so the areas of all bins sum to 1. In this way, histograms visualize how observations are distributed across the different values of the variable of interest. The age distribution for the survey respondents is right-skewed, suggesting that a larger number of young males were interviewed.

A histogram divides the data into bins where the area of each bin represents the proportion of observations that fall within the bin. The height of each bin represents **density**, which is equal to the proportion of observations within each bin divided by the width of the bin. A histogram approximates the distribution of a variable.

Our next histogram features the years of education variable, `educ.years`. Instead of letting R automatically choose the width of bins, as we did for the age variable, we now specify exactly how the bins are created using $[-0.5, 0.5), [0.5, 1.5), [1.5, 2.5), \ldots$, to center each bin around each of the integer values, i.e., $0, 1, 2, \ldots$, corresponding to the observed values. The height of each bin then represents the proportion of observations that received the corresponding number of years of education. We implement this by specifying a vector of the breakpoints between histogram bins with the `breaks` argument. In this case, the default specification, which we will get by leaving the argument unspecified, is $[0, 1), [1, 2), [2, 3), \ldots$ where it is centered around $0.5, 1.5, 2.5, \ldots$, failing to correspond to the observed values. Note that the `breaks` argument can take other forms of input to manipulate the histogram. For example, it also accepts a single integer specifying the number of bins for the histogram.

```
## histogram of education.  use "breaks" to choose bins
hist(afghan$educ.years, freq = FALSE,
     breaks = seq(from = -0.5, to = 18.5, by = 1),
     xlab = "Years of education",
     main = "Distribution of respondent's education")
## add a text label at (x, y) = (3, 0.5)
text(x = 3, y = 0.5, "median")
## add a vertical line representing median
abline(v = median(afghan$educ.years))
```

The histogram for the years of education variable clearly shows that the education level of these respondents is extremely low. Indeed, almost half of them have never attended school. We also add a vertical line and a text label indicating the *median* value, using the `abline()` and `text()` functions, respectively. Both of these functions add a layer to any existing plot, and this is why they are used after the `hist()` function in the above example. The `text(x,y,z)` function adds character text `z` centered at



the points specified by the coordinate vectors, (x, y). The `abline()` function can add a straight line to an existing plot in the following three ways:

- `abline(h=x)` to place a horizontal line at height `x`
- `abline(v=x)` to place a vertical line at point `x`
- `abline(a=y, b=s)` to place a line with intercept `y` and slope `s`

A more general function to plot a line is `lines()`. This function takes two arguments, `x` and `y`. These two arguments must be vectors with the same number of x-coordinates and y-coordinates respectively. The function will then draw line segments connecting the point denoted by the first coordinate in argument `x` and the first coordinate in argument `y`, to the point denoted by the second coordinates in each argument, to the point denoted by the third coordinates in each argument, and so on. For example, we can draw the median line as done above using this function instead.

```
## adding a vertical line representing the median
lines(x = rep(median(afghan$educ.years), 2), y = c(0,0.5))
```

In this example, we want to create a vertical line at the x value for the median of `afghan$educ.years`. We use y values, 0 and 0.5, so that the line will extend between the bottom and top limits of the histogram respectively. We then need x-coordinates equal to the median of `afghan$educ.years` to correspond with each y-coordinate. To do this easily, we can use the `rep()` function, whose first argument takes the value we want to repeat and whose second argument takes the number of repetitions, which is

the length of the resulting vector. The above `rep()` function creates a vector of length 2 with the median of `afghan$educ.years` as each element in that vector. Thus, a line goes from point $(x, y) = (1, 0)$ to point $(x, y) = (1, 0.5)$, since the median year of education is 1.

It is also possible to add points to any existing plot using the `points()` function. Specifically, in `points(x, y)`, two vectors—`x` and `y`—specify the coordinates of points to be plotted. Finally, R has various functionalities that enable users to choose different colors, line types, and other aesthetic choices. Some commonly used arguments are given below, but the details about each function can be obtained on their manual pages:

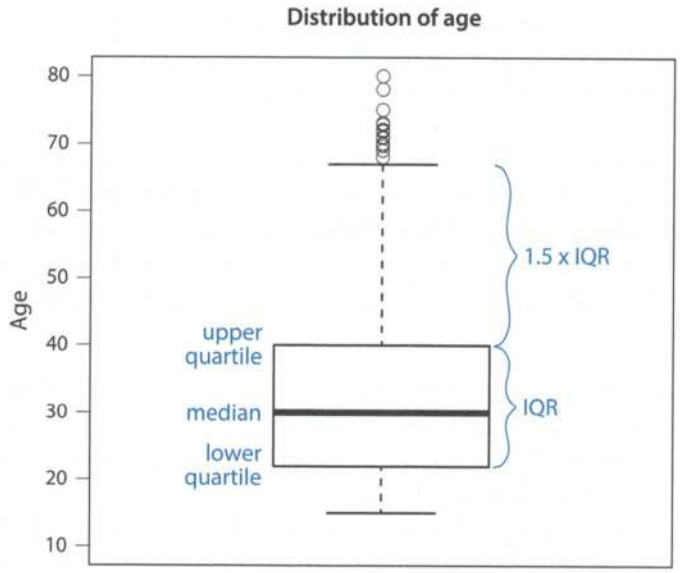
- `col` specifies the color to use, such as `"blue"` and `"red"`. This argument can be used in many functions including `text()`, `abline()`, `lines()`, and `points()`. Type `colors()` to see all the built-in color names R has (see section 5.3.3 for more details).
- `lty` specifies the type of line to be drawn, using either a character or a numeric value, including `"solid"` or `1` (default) for solid lines, `"dashed"` or `2` for dashed lines, `"dotted"` or `3` for dotted lines, `"dotdash"` or `4` for dotted and dashed lines, and `"longdash"` or `5` for long dashed lines. This argument can be used in many functions that produce lines, including `abline()` and `lines()`.
- `lwd` specifies the thickness of lines where `lwd = 1` is the default value. This argument can be used in many functions that produce lines, including `abline()` and `lines()`.

3.3.3 BOX PLOT

The *box plot* represents another way to visualize the distributions of a numeric variable. It is particularly useful when comparing the distribution of several variables by placing them side by side. A box plot visualizes the median, the quartiles, and the IQR all together as a single object. To make box plots in R, we use the `boxplot()` function by simply giving a variable of interest as an input. Again, we use the `age` variable as an example.

```
## commands for plotting curly braces and text in blue are omitted
boxplot(afghan$age, main = "Distribution of age", ylab = "Age",
        ylim = c(10, 80))
```

As illustrated below, the box contains 50% of the data ranging from the lower quartile (25th percentile) to the upper quartile (75th percentile) with the solid horizontal line indicating the median value (50th percentile). Then, dotted vertical lines, each of which has its end indicated by a short horizontal line called a "whisker," extend below and above the box. These two dotted lines represent the data that are contained within 1.5 IQR below the lower quartile and above the upper quartile, respectively. Furthermore, the observations that fall outside 1.5 IQR from the upper

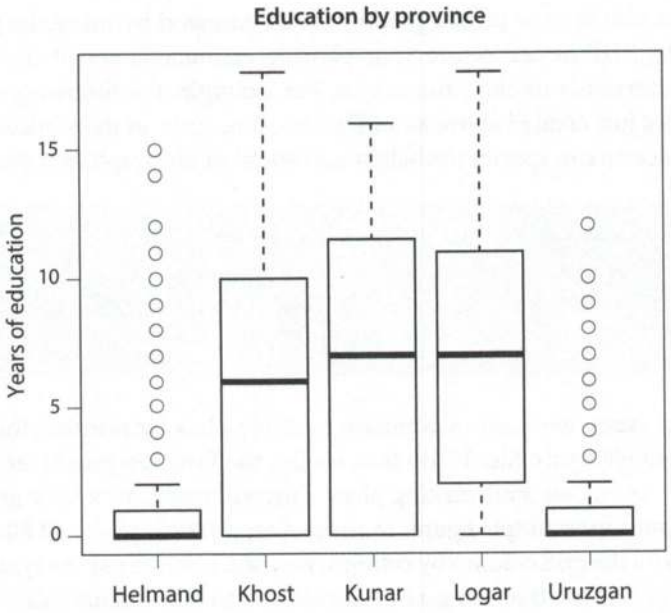


and lower quartiles are indicated by open circles. In this plot, the section of the dotted line extending from the top of the box to the horizontal line represents 1.5 IQR. If the minimum (maximum) value is contained within 1.5 IQR below the lower quartile (above the upper quartile), the dotted line will end at the minimum (maximum) value. The absence of open circles below the horizontal line implies that the minimum value of this variable is indeed within the 1.5 IQR of the lower quantile.

If we wish to visualize the distribution of a single variable, then a histogram is often more informative than a box plot because the former shows the full shape of the distribution. One of the main advantages of a box plot is that it allows us to compare multiple distributions in a more compact manner than histograms, as the next example shows. Using the `boxplot()` function, we can create a box plot for a different group of observations where the groups are defined by a factor variable. This is done by using the formula in R, which takes the form `y ~ x`. In the current context, `boxplot(y ~ x, data = d)` creates box plots for variable `y` for different groups defined by a factor `x` where the variables, `x` and `y`, are taken from the data frame `d`. As an illustration, we plot the distribution of the years of education variable by province.

```
boxplot(educ.years ~ province, data = afghan,
        main = "Education by province", ylab = "Years of education")
```

We find that the education level in Helmand and Uruzgan provinces is much lower than that of the other three provinces. It also turns out that civilians in these two provinces report harm inflicted by both parties more than those who live in the other provinces. This is shown below by computing the proportion of affirmative answers to the corresponding question, for each province.




```
tapply(afghan$violent.exp.taliban, afghan$province, mean, na.rm = TRUE)

##    Helmand      Khost      Kunar      Logar    Uruzgan
## 0.50422195 0.23322684 0.30303030 0.08024691 0.45454545

tapply(afghan$violent.exp.ISAF, afghan$province, mean, na.rm = TRUE)

##    Helmand     Khost     Kunar     Logar   Uruzgan
## 0.5410226 0.2424242 0.3989899 0.1440329 0.4960422
```

Note that the syntax, `na.rm = TRUE`, is passed to the `mean()` function within the `tapply()` function so that missing observations are deleted when computing the mean for each province (see section 3.2).

A **box plot** visualizes the distribution of a variable by indicating its median, lower and upper quartiles, and the points outside the 1.5 interquartile range from the lower and upper quartiles. It enables the comparison of distributions across multiple variables in a compact manner.

3.3.4 PRINTING AND SAVING GRAPHS

There are a few ways to print and save the graphs you create in R. The easiest way is to use the menus in RStudio. In RStudio, each time you create a graph using any of the R plotting functions, a new tab will open in the bottom-right window. To save an image of the plot, click `Export` and then either `Save Plot as Image` or `Save Plot as PDF`.

You can also save or print a graph with a command by using the `pdf()` function to open the PDF device before your plotting commands and then the `dev.off()` function afterwards to close the device. For example, the following syntax saves the box plots we just created above as a PDF file `educ.pdf` in the working directory. The `pdf()` function can specify the height and width of the graphics region in inches.

```
pdf(file = "educ.pdf", height = 5, width = 5)
boxplot(educ.years ~ province, data = afghan,
        main = "Education by province", ylab = "Years of education")
dev.off()
```

In many cases, we want to compare multiple plots by printing them next to each other in a single figure file. To do this, we use the function `par()` as `par(mfrow = c(X, Y))` before we start making plots. This will create an `X` by `Y` grid of "subplots" (`mfrow` stands for multiple figures in rows). Our multiple plots will fill in this grid, row by row. To fill the grid column by column, you can, instead, use the syntax `par(mfcol = c(X, Y))`. Note that the `par()` function also takes many other arguments that allow users to control graphics in R. For example, the `cex` argument changes the size of a character or symbol, with `cex = 1` as the default value. We can set the `cex` argument to a value greater than 1 (e.g., `par(cex = 1.2)`) in order to enlarge the fonts in displayed graphics. Note that it is also possible to separately specify the size for different parts of a plot using `cex.main` (main plot title), `cex.lab` (axis title labels), and `cex.axis` (axis value labels). Executing the following code chunk all at once creates the two histograms we made earlier in this chapter and saves them side by side in a single PDF file.

```
pdf(file = "hist.pdf", height = 4, width = 8)
## one row with 2 plots with font size 0.8
par(mfrow = c(1, 2), cex = 0.8)
## for simplicity omit the text and lines from the earlier example
hist(afghan$age, freq = FALSE,
     xlab = "Age", ylim = c(0, 0.04),
     main = "Distribution of respondent's age")
hist(afghan$educ.years, freq = FALSE,
     breaks = seq(from = -0.5, to = 18.5, by = 1),
     xlab = "Years of education", xlim = c(0, 20),
     main = "Distribution of respondent's education")
dev.off()
```

3.4 Survey Sampling

Survey sampling is one of the main data collection methods in quantitative social science research. It is often used to study public opinion and behavior when such

information is not available from other sources such as administrative records. Survey sampling is a process in which researchers select a subset of the population, called a sample, to understand the features of a target population. It should be distinguished from a *census*, for which the goal is to enumerate all members of the population.

What makes survey sampling remarkable is that one can learn about a fairly large population by interviewing a small fraction of it. In the Afghanistan data, a sample of 2754 respondents was used to infer the experiences and attitudes of approximately 15 million civilians. In the United States, a sample of just about 1000 respondents is typically used to infer the public opinion of more than 200 million adult citizens. In this section, we explain what makes this seemingly impossible task possible and discuss important methodological issues when collecting and analyzing survey data.

3.4.1 THE ROLE OF RANDOMIZATION

As in the randomized control trials (RCTs) discussed in chapter 2, randomization plays an essential role in survey sampling. We focus on a class of sampling procedures called *probability sampling* in which every unit of a target population has a known nonzero probability of being selected. Consider the most basic probability sampling procedure, called *simple random sampling* (SRS), which selects the predetermined number of respondents to be interviewed from a target population, with each potential respondent having an equal chance of being selected. The sampling is done *without replacement* rather than *with replacement* so that once individuals are selected for interview they are taken out of the *sampling frame*, which represents the complete list of potential respondents. Therefore, sampling without replacement assigns at most one interview per individual.

SRS produces a sample of respondents that are *representative* of the population. By "representative," we mean that if we repeat the procedure many times, the features of each resulting sample would not be exactly the same as those of the population, but on average (across all the samples) would be identical. For example, while one may happen to obtain, due to random chance, a sample of individuals who are slightly older than those of the population, the age distribution over repeated samples would resemble that of the population. Moreover, as in RCTs, probability sampling guarantees that the characteristics of the sample, whether observed or unobserved, are on average identical to the corresponding characteristics of the population. For this reason, we can infer population characteristics using those of a representative sample obtained through probability sampling procedures (see chapter 7 for more details).

Before probability sampling was invented, researchers often used a procedure called *quota sampling*. Under this alternative sampling strategy, we specify fixed quotas of certain respondents to be interviewed such that the resulting sample characteristics resemble those of the population. For example, if 20% of the population has a college degree, then researchers will set the maximum number of college graduates who will be selected for interview to be 20% of the sample size. They will stop interviewing those with college degrees once they reach that quota. The quota can be defined using multiple variables. Often, the basic demographics such as age, gender, education, and race are used to construct the categories for which the quota is specified. For example, we may interview black females with a college degree and between 30 and 40 years old, up to 5% of the sample size.

The problem of quota sampling is similar to that of the observational studies discussed in chapter 2. Even if a sample is representative of the population in terms of some observed characteristics, which are used to define quotas, its unobserved features may be quite different from those of the population. Just as individuals may self-select to receive a treatment in an observational study, researchers may inadvertently interview individuals who have characteristics systematically different from those who are not interviewed. Probability sampling eliminates this potential *sample selection bias* by making sure that the resulting sample is representative of the target population.

Simple random sampling (SRS) is the most basic form of **probability sampling**, which avoids **sample selection bias** by randomly choosing units from a population. Under SRS, the predetermined number of units is randomly selected from a target population without replacement, where each unit has an equal probability of being selected. The resulting sample is representative of the population in terms of any observed and unobserved characteristics.

Quota sampling is believed to have caused one of the most well-known errors in the history of newspapers. In the 1948 US presidential election, most major preelection polls, including those conducted by Gallup and Roper, used quota sampling and predicted that Thomas Dewey, then the governor of New York, would decisively defeat Harry Truman, the incumbent, on Election Day. On election night, the *Chicago Tribune* went ahead and sent the next morning's newspaper to press, with the erroneous headline "Dewey defeats Truman," even before many East Coast states reported their polling results. The election result, however, was the exact opposite. Truman won by a margin of 5 percentage points in the national vote. Figure 3.1 shows a well-known picture of Truman happily holding a copy of the *Chicago Tribune* with the erroneous headline.

In order to apply SRS, we need a list of all individuals in the population to sample from. As noted earlier, such a list is called a *sampling frame*. In practice, given a target population, obtaining a sampling frame that enumerates all members of the population is not necessarily straightforward. Lists of phone numbers, residential addresses, and email addresses are often incomplete, missing a certain subset of the population who have different characteristics. *Random digit dialing* is a popular technique for phone surveys. However, the procedure may suffer from sample selection bias since some people may not have a phone number and others may have multiple phone numbers.

Most in-person surveys employ a complex sampling procedure due to logistical challenges. While an in-depth study of various survey sampling strategies is beyond the scope of this book, we briefly discuss how the Afghanistan survey was conducted in order to illustrate how survey sampling is done in practice. For the Afghanistan survey, the researchers used a *multistage cluster sampling* procedure. In countries like Afghanistan, it is difficult to obtain a sampling frame that contains most, let alone all, of their citizens. However, comprehensive lists of administrative units such as districts and villages are often readily available. In addition, since sending interviewers across a large number of distant areas may be too costly, it is often necessary to sample respondents within a reasonable number of subregions.

Figure 3.1. Harry Truman, the Winner of the 1948 US Presidential Election, Holding a Copy of the *Chicago Tribune* with the Erroneous Headline. Source: Copyright unknown, Courtesy of Harry S. Truman Library.

Table 3.2. Afghanistan Village Data.

Variable	*Description*
`village.surveyed`	whether a village is sampled for survey
`altitude`	altitude of the village
`population`	population of the village

The multistage cluster sampling method proceeds in multiple stages by sampling larger units first and then randomly selecting smaller units within each of the selected larger units. In the Afghanistan survey, within each of the five provinces of interest, the researchers sampled districts and then villages within each selected district. Within each sampled village, interviewers selected a household in an approximately random manner based on their location within the village, and finally administered a survey to a male respondent aged 16 years or older, who was sampled using the *Kish grid* method. While the probability of selecting each individual in the population is known only approximately, the method in theory should provide a roughly representative sample of the target population.

We examine the representativeness of the randomly sampled villages in the Afghanistan data. The data file `afghan-village.csv` contains the altitude and population of each village (see table 3.2 for the names and descriptions of the variables). For the population variable, it is customary to take the *logarithmic transformation* so that the distribution does not look too skewed with a small number of extremely large or small values. The logarithm of a positive number x is defined as the exponent of a base value b, i.e., $y = \log_b x \iff x = b^y$. For example, if the base value is 10, then the

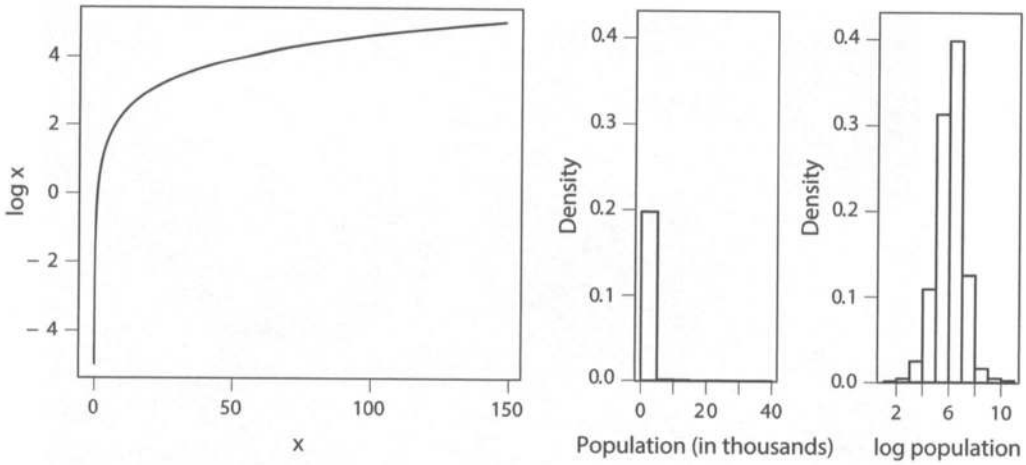


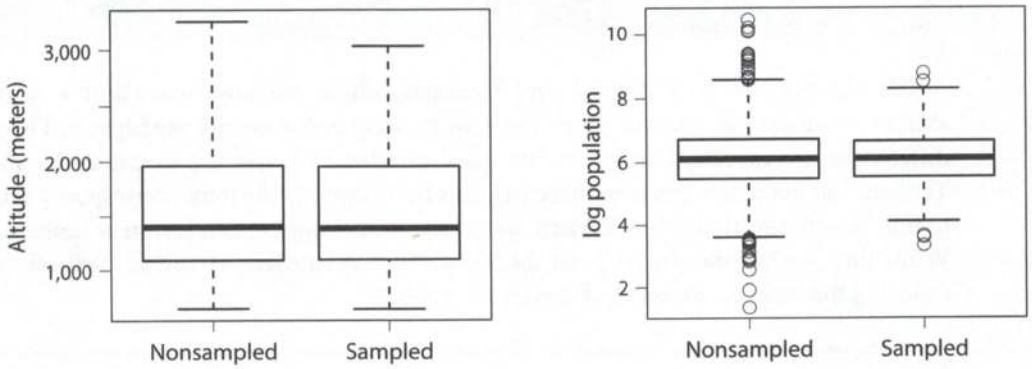
Figure 3.2. The Natural Logarithm. The left plot shows the natural logarithm $\log_e x$ where x is a positive number and $e = 2.7182\ldots$ is Euler's number. The remaining plots display the histograms for the population of Afghan villages on the original scale (in thousands) and the natural logarithmic scale. The population distribution is skewed without the logarithmic transformation.

logarithm of 1000 is $3 = \log_{10} 1000$. Similarly, the logarithm of 0.01 is $-2 = \log_{10} 0.01$. The *natural logarithm* uses as its base value an important mathematical constant $e = 2.7182\ldots$, which is defined as the limit of $(1 + 1/n)^n$ as n approaches infinity and is sometimes called Euler's number, so that $y = \log_e x \iff x = e^y$. The left-hand plot of figure 3.2 depicts the natural logarithm function graphically. The figure also shows that in the Afghanistan data, without the logarithmic transformation, the distribution of the population is quite skewed because there exist a large number of small villages and a small number of large villages.

The **natural logarithmic transformation** is often used to correct the skewness of variables such as income and population that have a small number of observations with extremely large or small positive values. The natural logarithm is the logarithm with base e, which is a mathematical constant approximately equal to 2.7182, and defined as $y = \log_e x$. It is the inverse function of the exponential function, so $x = e^y$.

We use box plots to compare the distribution of these variables across sampled and nonsampled villages. The variable `village.surveyed` indicates whether each village in the data is (randomly) sampled and surveyed; `1` indicates yes and `0` no. As explained above, we take the natural logarithmic transformation for the population variable using the `log()` function. By default R uses e as its base, though it is possible to specify a different base using the `base` argument in this function. Note that the exponential function in R is given by `exp()`. In the `boxplot()` function, we can use the `names` argument to specify a character vector of labels for each group.

```
## load village data
afghan.village <- read.csv("afghan-village.csv")
## box plots for altitude
boxplot(altitude ~ village.surveyed, data = afghan.village,
        ylab = "Altitude (meters)",  names = c("Nonsampled", "Sampled"))
## box plots for log population
boxplot(log(population) ~ village.surveyed, data = afghan.village,
        ylab = "log population", names = c("Nonsampled", "Sampled"))
```



The result shows that although there are some outliers, the distribution of these two variables is largely similar between the sampled and nonsampled villages. So, at least for these variables the sample appears to be representative of the population.

3.4.2 NONRESPONSE AND OTHER SOURCES OF BIAS

While probability sampling has attractive theoretical properties, in practice conducting a survey faces many obstacles. As mentioned earlier, a *sampling frame*, which enumerates all members of a target population, is difficult to obtain. In many cases, we end up sampling from a list that may systematically diverge from the target population in terms of some important characteristics. Even if a representative sampling frame is available, interviewing randomly selected individuals may not be straightforward. Failure to reach selected units is called *unit nonresponse*. For example, many individuals refuse to participate in phone surveys. In the Afghanistan survey, the authors report that 2754 out of 3097 potential respondents agreed to participate in the survey, resulting in an 11% refusal rate. If those to whom researchers fail to administer the survey are systematically different from those who participate in the survey, then bias due to unit nonresponse arises.

In addition to unit nonresponse, most surveys also encounter the *item nonresponse* problem when respondents refuse to answer certain survey questions. For example, we saw in section 3.2 that in the Afghanistan survey, the income variable had a nonresponse rate of approximately 5%. If those who refuse to answer are systematically different from those who answer, then the resulting inference based only on the

observed responses may be biased. In the Afghanistan data, for example, the item nonresponse rates for the questions about civilian victimization by the Taliban and the ISAF appear to vary across provinces.

```
tapply(is.na(afghan$violent.exp.taliban), afghan$province, mean)
##     Helmand       Khost       Kunar       Logar     Uruzgan
## 0.030409357 0.006349206 0.000000000 0.000000000 0.062015504

tapply(is.na(afghan$violent.exp.ISAF), afghan$province, mean)
##     Helmand       Khost       Kunar       Logar     Uruzgan
## 0.016374269 0.004761905 0.000000000 0.000000000 0.020671835
```

We observe that in Helmand and Uruzgan, which are known to be the most violent provinces (see section 3.3.3), the item nonresponse rates are the highest. These differences are especially large for the question about civilian victimization by the Taliban. The evidence presented here suggests that although the item nonresponse rate in this survey is relatively low, certain systematic factors appear to affect its magnitude. While they are beyond the scope of this book, there exist many statistical methods of reducing the bias due to unit and item nonresponse.

There are two types of nonresponse in survey research. **Unit nonresponse** refers to a case in which a potential respondent refuses to participate in a survey. **Item nonresponse** occurs when a respondent who agreed to participate refuses to answer a particular question. Both nonresponses can result in biased inferences if those who respond to a question are systematically different from those who do not.

Beyond item and unit nonresponse, another potential source of bias is *misreporting*. Respondents may simply lie because they may not want interviewers to find out their true answers. In particular, *social desirability bias* refers to the problem where respondents choose an answer that is seen as socially desirable regardless of what their truthful answer is. For example, it is well known that in advanced democracies voters tend to report they participated in an election even when they actually did not, because abstention is socially undesirable. Similarly, social desirability bias makes it difficult to accurately measure sensitive behavior and opinions such as corruption, illegal behavior, racial prejudice, and sexual activity. For this reason, some scholars remain skeptical of self-reports as measurement for social science research.

One main goal of the Afghanistan study was to measure the extent to which Afghan citizens support foreign forces. To defeat local insurgent forces and win the wars in Afghanistan and Iraq, many Western policy makers believed that "winning the hearts and minds" of a civilian population was essential. Unfortunately, directly asking whether citizens are supportive of foreign forces and insurgents in rural Afghan

villages can put interviewers and respondents at risk because interviews are often conducted in public. The *Institutional Review Board*, which evaluates the ethical issues and potential risks of research projects involving human subjects, may not approve direct questioning of sensitive questions in a civil war setting. Even if possible, direct questioning may lead to nonresponse and misreporting.

To address this problem, the authors of the original study implemented a survey methodology called *item count technique* or *list experiment*. The idea is to use aggregation to provide a certain level of anonymity to respondents. The method first randomly divides the sample into two comparable groups. In the "control" group, the following question was asked.

> I'm going to read you a list with the names of different groups and individuals on it. After I read the entire list, I'd like you to tell me how many of these groups and individuals you broadly support, meaning that you generally agree with the goals and policies of the group or individual. Please don't tell me which ones you generally agree with; only tell me how many groups or individuals you broadly support.
>
> Karzai Government; National Solidarity Program; Local Farmers

The "treatment" group received the same question except with an additional sensitive item:

> Karzai Government; National Solidarity Program; Local Farmers; Foreign Forces

Here, the last item, Foreign Forces, which refers to the ISAF, is the sensitive item. The item count technique does not require respondents to answer each item separately. Instead, they give an aggregate count of items. Since the two conditions are comparable apart from the sensitive item, the difference in the average number of items a respondent reports will be an estimate of the proportion of those who support the ISAF. The `list.group` variable indicates which group each respondent is randomly assigned to, where for the two relevant groups the variable equals `ISAF` and `control`. The outcome variable is `list.response`, which represents the item count reported by each respondent.

```
mean(afghan$list.response[afghan$list.group == "ISAF"])-
    mean(afghan$list.response[afghan$list.group == "control"])
## [1] 0.04901961
```

The item count technique estimates that approximately 5% of Afghan citizens support the ISAF, implying that the ISAF is unpopular among Afghans.

The weakness of the item count technique, however, is that in the "treatment" group, answering either "0" or "4" in this case reveals one's honest answer. These potential problems are called *floor effects* and *ceiling effects*, respectively. In the Afghan data, we see clear evidence of this problem when the Taliban, instead of the ISAF, is added to the list as the sensitive item.

```
table(response = afghan$list.response, group = afghan$list.group)
##          group
## response control ISAF taliban
##        0     188  174       0
##        1     265  278     433
##        2     265  260     287
##        3     200  182     198
##        4       0   24       0
```

Remarkably, no respondents in the `taliban` group answered either "0" or "4," perhaps because they do not want to be identified as either supportive or critical of the Taliban.

As we can see, measuring the truthful responses to sensitive questions is a challenging task. In addition to the item count technique, social scientists have used a variety of survey methodologies in an effort to overcome this problem. Another popular methodology is called the *randomized response technique* in which researchers use randomization to provide anonymity to respondents. For example, respondents are asked to roll a six-sided die in private without revealing the outcome. They are then asked to answer yes if the outcome of rolling the die was 1, no if 6, and give an honest answer if the outcome was between 2 and 5. Therefore, unlike the item count technique, the secrecy of individual responses is completely protected. Since the probability of each outcome is known, the researchers can estimate the aggregate proportion of honest responses out of those who responded with a yes answer even though they have no way of knowing the truthfulness of individual answers with certainty.

3.5 Measuring Political Polarization

Social scientists often devise *measurement models* to summarize and understand the behaviors, attitudes, and unobservable characteristics of human beings. A prominent example is the question of how to quantitatively characterize the *ideology* of political actors such as legislators and judges from their behavior. Of course, we do not directly observe the extent to which an individual is liberal or conservative. While ideology is perhaps a purely artificial concept, it is nonetheless a useful way to describe the political orientation of various individuals. Over the past several decades, social scientists have attempted to infer the ideology of politicians from their roll call votes. In each year, for example, legislators in the US Congress vote on hundreds of bills. Using this voting record, which is publicly available, researchers have tried to characterize the political ideology of each member of Congress and how the overall ideological orientation in the US Congress has changed over time.[2]

A simple measurement model of *spatial voting* can relate a legislator's ideology to their votes. Figure 3.3 illustrates this model, which characterizes the ideology

[2] This section is based on Nolan McCarty, Keith T. Poole, and Howard Rosenthal (2006) *Polarized America: The Dance of Ideology and Unequal Riches*. MIT Press.

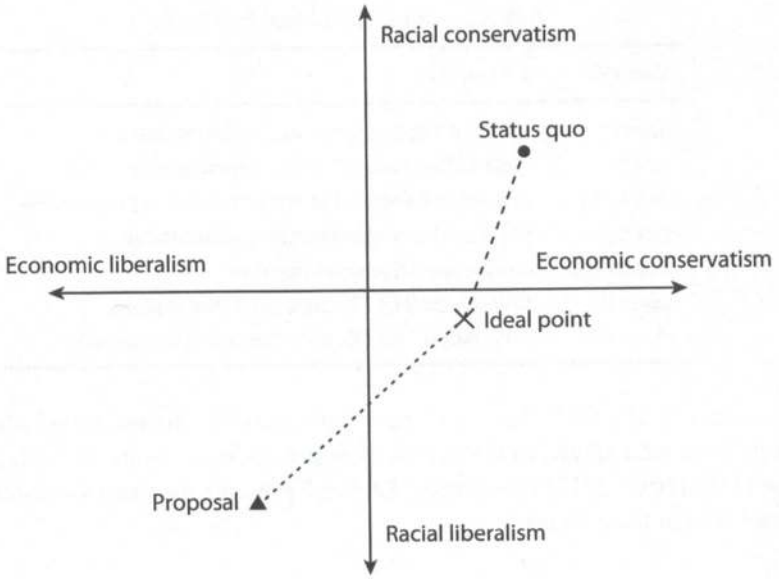


Figure 3.3. An Illustration for the Spatial Voting Model of Legislative Ideology.

or "ideal point" of legislators by two dimensions—economic and racial liberalism/conservatism—identified by researchers as the main ideological characteristics of postwar congressional politics. Researchers have found that much of congressional roll call voting can be explained by the economic liberalism/conservatism dimension while the racial liberalism/conservatism dimension is less pronounced. Under this model, the legislator, whose ideal point is indicated by a cross mark in the figure, is more likely to vote against the proposal (solid triangle) whenever their ideal point is closer to the status quo (solid circle) than to the proposal location. The outcomes of congressional votes on controversial proposals reveal much about legislators' ideologies. On the other hand, a unanimously accepted or rejected proposal provides no information about legislators' ideological orientations.

A similar model is used in educational testing literature. Scholars have developed a class of statistical methods called *item response theory* for standardized tests such as the SAT and Graduate Record Examination (GRE). In this context, legislators and legislative proposals are replaced with student examinees and exam questions. Instead of ideal points, the goal is to measure students' abilities. The model also estimates the difficulty of each question. This helps the researchers choose good exam questions, which are neither too difficult nor too easy, so that only competent students will be able to provide a correct answer. These examples illustrate the importance of latent (i.e., unobserved) measurements in social science research.

3.6 Summarizing Bivariate Relationships

In this section, we introduce several ways to summarize the relationship between two variables. We analyze the estimates of legislators' ideal points, known as *DW-NOMINATE scores*, where more negative (positive) scores are increasingly liberal

Table 3.3. Legislative Ideal Points Data.

Variable	*Description*
`name`	name of the congressional representative
`state`	state of the congressional representative
`district`	district number of the congressional representative
`party`	party of the congressional representative
`congress`	congressional session number
`dwnom1`	DW-NOMINATE score (first dimension)
`dwnom2`	DW-NOMINATE score (second dimension)

(conservative). The CSV file `congress.csv` contains the estimated ideal points of all legislators who served in the House of Representatives from the 80th (1947–1948) to the 112th (2011–2012) Congresses. Table 3.3 presents the names and descriptions of the variables in the data set.

3.6.1 SCATTER PLOT

Using the `plot()` function, we create a *scatter plot*, which plots one variable against another in order to visualize their relationship. The syntax for this function is `plot(x, y)`, where `x` and `y` are vectors of horizontal and vertical coordinates, respectively. Here, we plot the DW-NOMINATE first dimension score (`dwnom1` variable) on the horizontal axis, which represents economic liberalism/conservatism, against its second dimension score on the vertical axis (`dwnom2` variable), which represents racial liberalism/conservatism. We will start by creating scatter plots for the 80th and 112th Congresses. We begin by subsetting the relevant part of the data.

```
congress <- read.csv("congress.csv")
## subset the data by party
rep <- subset(congress, subset = (party == "Republican"))
dem <- congress[congress$party == "Democrat", ] # another way to subset
## 80th and 112th Congress
rep80 <- subset(rep, subset = (congress == 80))
dem80 <- subset(dem, subset = (congress == 80))
rep112 <- subset(rep, subset = (congress == 112))
dem112 <- subset(dem, subset = (congress == 112))
```

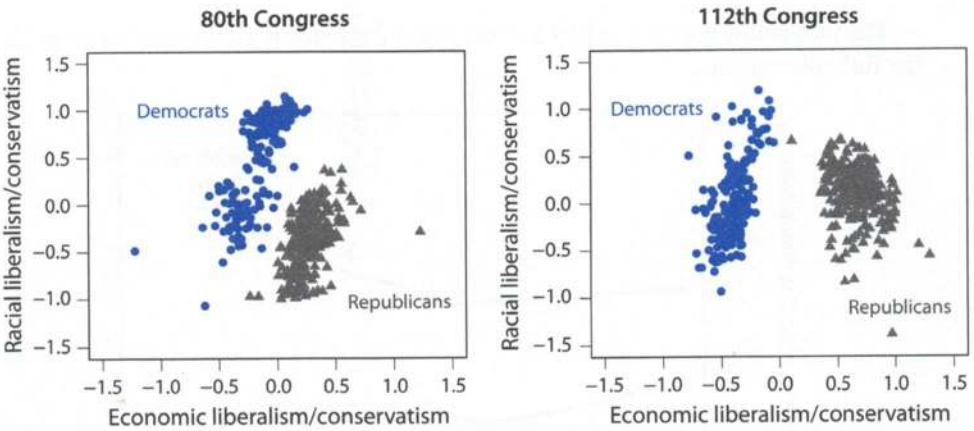
We will be creating multiple scatter plots with the same set of axis labels and axis limits. To avoid repetition, we store them as objects for later use.

```
## preparing the labels and axis limits to avoid repetition
xlab <- "Economic liberalism/conservatism"
ylab <- "Racial liberalism/conservatism"
lim <- c(-1.5, 1.5)
```

Finally, using this axis information, we create scatter plots of ideal points for the 80th and 112th Congresses. Note that the `pch` argument in the `plot()` and `points()` functions can be used to specify different plotting symbols for the two parties. In the current example, `pch = 16` graphs solid triangles for Republicans while `pch = 17` graphs solid circles for Democrats. More options are available and can be viewed by typing `example(points)` into R console.

```
## scatter plot for the 80th Congress
plot(dem80$dwnom1, dem80$dwnom2, pch = 16, col = "blue",
     xlim = lim, ylim = lim, xlab = xlab, ylab = ylab,
     main = "80th Congress") # Democrats
points(rep80$dwnom1, rep80$dwnom2, pch = 17, col = "red") # Republicans
text(-0.75, 1, "Democrats")
text(1, -1, "Republicans")
## scatter plot for the 112th Congress
plot(dem112$dwnom1, dem112$dwnom2, pch = 16, col = "blue",
     xlim = lim, ylim = lim, xlab = xlab, ylab = ylab,
     main = "112th Congress")
points(rep112$dwnom1, rep112$dwnom2, pch = 17, col="red")
```

The plots below use solid gray triangles instead of red triangles for Republicans. See page C1 for the full-color version.



The plots show that in the 112th Congress (as opposed to the 80th Congress), the racial liberalism/conservatism dimension is no longer important in explaining the ideological difference between Democrats and Republicans. Instead, the economic dimension appears to be a dominant explanation for the partisan difference, and the difference between Democrats and Republicans in the racial dimension is much less pronounced.

Next, we compute the median legislator, based on the DW-NOMINATE first dimension score, separately for the Democratic and Republican Parties and for each Congress. These party median ideal points represent the center of each party in the

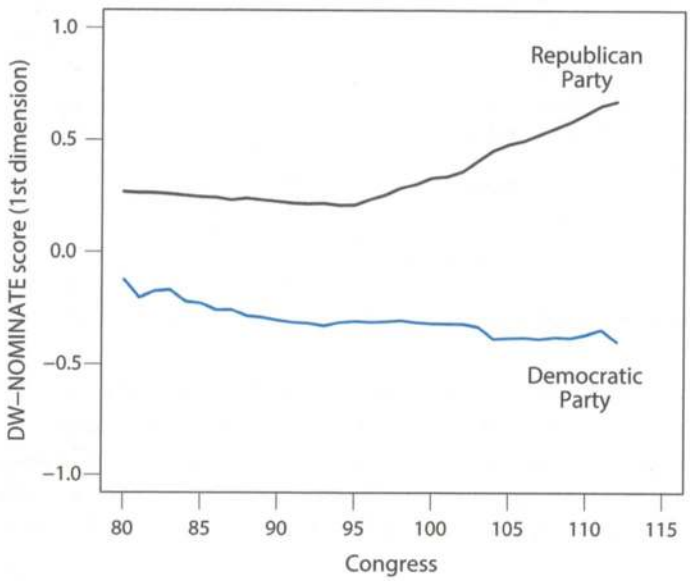
economic liberalism/conservatism dimension. We can do this easily by using the `tapply()` function.

```
## party median for each congress
dem.median <- tapply(dem$dwnom1, dem$congress, median)
rep.median <- tapply(rep$dwnom1, rep$congress, median)
```

Finally, using the `plot()` function, we create a *time-series plot* where each party median is displayed for each Congress. We set the `type` argument to `"l"` in order to draw a line connecting the median points over time. This plot enables us to visualize how the party medians have changed over time. We will use the term of Congress as the horizontal axis. This information is available as the `name` of the `dem.median` vector.

```
## Democrats
plot(names(dem.median), dem.median, col = "blue", type = "l",
     xlim = c(80, 115), ylim = c(-1, 1), xlab = "Congress",
     ylab = "DW-NOMINATE score (first dimension)")
## add Republicans
lines(names(rep.median), rep.median, col = "red")
text(110, -0.6, "Democratic\n Party")
text(110, 0.85, "Republican\n Party")
```

The plot below uses a gray line instead of a red line for Republicans. See page C1 for the full-color version.



Note that the syntax `\n` used in the `text()` function indicates a change to a new line. The plot clearly shows that the ideological centers of the two parties diverge over

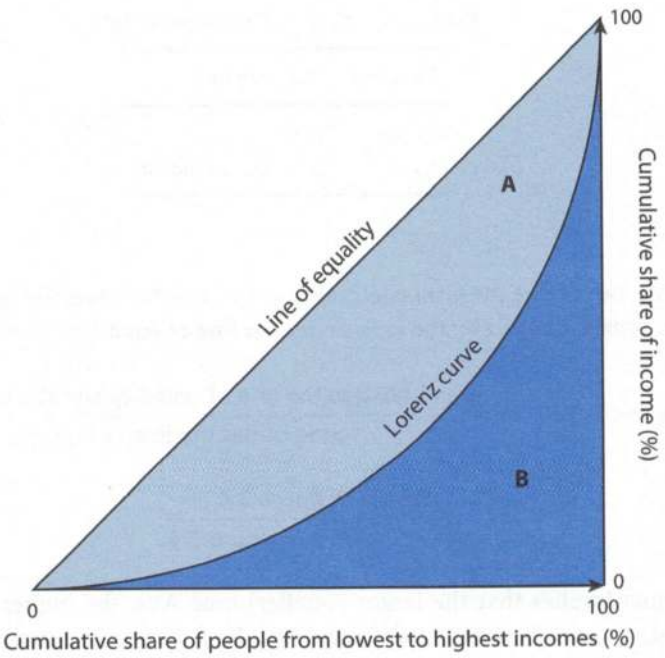


Figure 3.4. Gini Coefficient and Lorenz Curve.

time. The Democratic Party has become more liberal while the Republican Party has increasingly moved in a conservative direction in recent years. Many scholars refer to this phenomenon as *political polarization*.

A **scatter plot** graphically compares two variables measured on the same set of units by plotting the value of one variable against that of the other for each unit.

3.6.2 CORRELATION

What is the cause of political polarization? This is a difficult question to answer, and is the subject of much scholarly debate. However, it has been pointed out that rising income inequality may be responsible for the widening partisan gap. To measure income inequality, we use the *Gini coefficient* (*Gini index*), which is best understood graphically. Figure 3.4 illustrates the idea. The horizontal axis represents the cumulative share of people sorted from the lowest to highest income. The vertical axis, on the other hand, plots the cumulative share of income held by those whose income is equal to or less than that of a person at a given income percentile. The *Lorenz curve* connects these two statistics. If everyone earns exactly the same income, then the Lorenz curve will be the same as the 45-degree line because x% of the population will hold exactly x% of national income regardless of the value of x. Let's call this the line of equality. However, if low income people earn a lot less than high income people, the Lorenz curve will become flatter at the beginning and then sharply increase at the end.

Table 3.4. US Gini Coefficient Data.

Variable	*Description*
`year`	year
`gini`	US Gini coefficient

Now, we can define the Gini coefficient as the area between the line of equality and the Lorenz curve divided by the area under the line of equality. In terms of figure 3.4,

$$\text{Gini coefficient} = \frac{\text{area between the line of equality and the Lorenz curve}}{\text{area under the line of equality}}$$

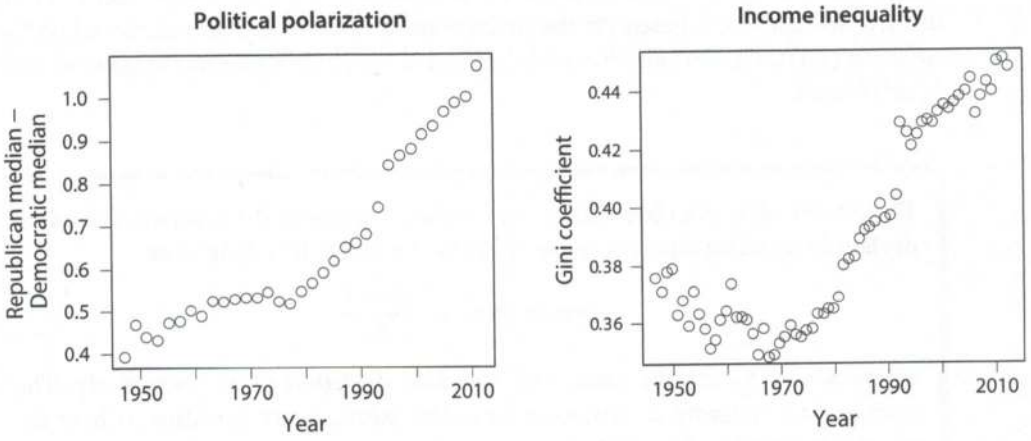
$$= \frac{\text{area A in figure 3.4}}{\text{area A + area B in figure 3.4}}.$$

The formula implies that the larger (smaller) area A is, the higher (lower) the Gini coefficient, meaning more (less) inequality. In a perfectly equal society, the Gini coefficient is 0. In contrast, a society where one person possesses all the wealth has a Gini coefficient of 1.

The **Gini coefficient** (Gini index) measures the degree of income equality and inequality in a given society. It ranges from 0 (everyone has the same amount of wealth) to 1 (one person possesses all the wealth).

To examine the relationship between political polarization and income inequality, we create two time-series plots side by side. The first plot shows the partisan gap, i.e., the difference between the two party medians, over time. The second time-series plot displays the Gini coefficient during the same time period. The CSV data file, `USGini.csv`, contains the Gini coefficient from 1947 to 2013 (see table 3.4). We notice that both political polarization and income inequality have been steadily increasing in the United States.

```
## Gini coefficient data
gini <- read.csv("USGini.csv")
## time-series plot for partisan difference
plot(seq(from = 1947.5, to = 2011.5, by = 2),
     rep.median - dem.median, xlab = "Year",
     ylab = "Republican median -\n Democratic median",
     main = "Political polarization")
## time-series plot for Gini coefficient
plot(gini$year, gini$gini, ylim = c(0.35, 0.45), xlab = "Year",
     ylab = "Gini coefficient", main = "Income inequality")
```



However, in chapter 2, we learned that *association does not necessarily imply causation* and hence we should not necessarily interpret this upwards trend as evidence for income inequality causing polarization. For example, life expectancy has also constantly increased during this time period, and yet this does not imply that longer life expectancy caused political polarization or vice versa.

Correlation (also referred to as a *correlation coefficient*) is one of the most frequently used statistics to summarize bivariate relationships. The measure represents how, on average, two variables move together relative to their respective means. Before defining correlation, we need to introduce the *z-score*, which represents the number of standard deviations an observation is above or below the mean. Specifically, the z-score of the ith observation of variable x is defined as

$$z\text{-score of } x_i = \frac{x_i - \text{mean of } x}{\text{standard deviation of } x}. \tag{3.1}$$

For example, if the z-score of a particular observation equals 1.5, the observation is 1.5 standard deviations above the mean. The z-score standardizes a variable so its unit of measurement no longer matters. More formally, the z-score of $ax_i + b$, where a and b are constants (a is non-zero), is identical to the z-score of x_i. Simple algebra can show this property:

$$\begin{aligned} z\text{-score of } (ax_i + b) &= \frac{(ax_i + b) - \text{mean of } (ax + b)}{\text{standard deviation of } (ax + b)} \\ &= \frac{a \times (x_i - \text{mean of } x)}{a \times \text{standard deviation of } x} \\ &= z\text{-score of } x_i, \end{aligned}$$

where the first equality follows from the definition of z-score in equation (3.1) and the second equality is based on the definitions of mean and standard deviation (see equation (2.4)). The constant b can be dropped in the above equations because its mean equals b itself.

The z-**score** of the ith observation of a variable x measures the number of standard deviations an observation is above or below the mean. It is defined as

$$z\text{-score of } x_i = \frac{x_i - \bar{x}}{S_x},$$

where $\bar{x}$ and S_x are the mean and standard deviation of x, respectively. The z-score, as a measure of deviation from the mean, is not sensitive to how the variable is scaled and/or shifted.

Now, we can define the correlation between two variables x and y, measured for the same set of n observations, as the average products of z-scores for the two variables:

$$\text{correlation}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(z\text{-score of } x_i \times z\text{-score of } y_i). \tag{3.2}$$

As in the case of standard deviation (see section 2.6.2), the denominator of the correlation is often $n - 1$ rather than n. However, this difference should not affect one's conclusion so long as the sample size is sufficiently large. Within the summation, each z-score measures the deviation of the corresponding observation from its mean in terms of standard deviation. Suppose that when one variable is above its mean, the other variable is also likely to be greater than its own mean. Then, the correlation is likely to be positive because the signs of the standardized units tend to agree with each other. On the other hand, suppose that when one variable is above its mean, the other variable is likely to be less than its own mean. Then, the correlation is likely to be negative. In the current example, a positive correlation means that in years when income inequality is above its over-time mean, political polarization is also likely to be higher than its over-time mean.

Recall that z-scores are not sensitive to what units are used to measure a variable. Because it is based on z-scores, correlation also remains identical even if different units are used for measurement. For example, the correlation does not change even if one measures income in thousands of dollars instead of dollars. Indeed, one can even use a different currency. This is convenient because, for example, the relationship between income and education should not change depending on what scales we use to measure income. As another consequence of standardization, correlation varies only between -1 and 1. This allows us to compare the strengths and weaknesses of association between different pairs of variables.

Correlation (correlation coefficient) measures the degree to which two variables are associated with each other. It is defined as

$$\text{correlation of } x \text{ and } y = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{S_x} \times \frac{y_i - \bar{y}}{S_y}\right)$$

$$\text{or} \quad \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{S_x} \times \frac{y_i - \bar{y}}{S_y}\right),$$

where $\bar{x}$ and $\bar{y}$ are the means and S_x and S_y are the standard deviations for variables x and y, respectively. Correlation ranges from -1 to 1 and is not sensitive to how a variable is scaled and/or shifted.

In R, the correlation can be calculated using the `cor()` function. For example, we can now calculate the correlation between the Gini coefficient and the measure of political polarization. To do this, since each US congressional session lasts two years, we take the Gini coefficient for the second year of each session.

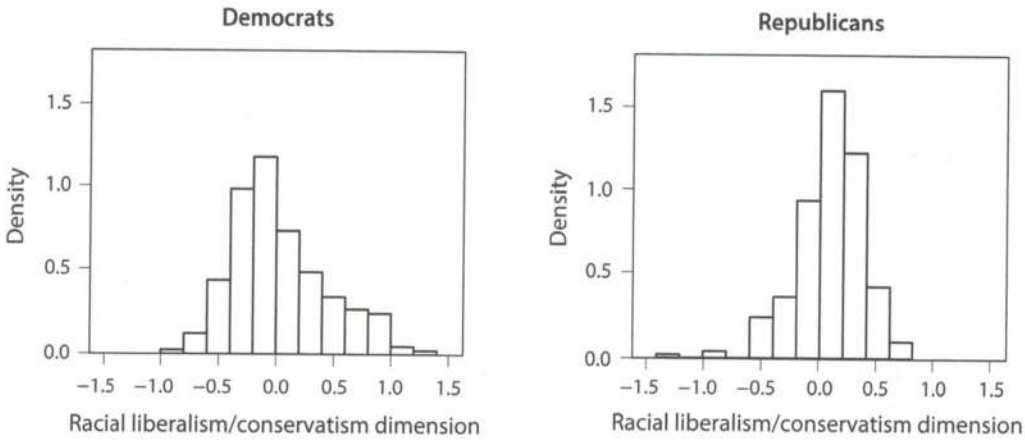
```
cor(gini$gini[seq(from = 2, to = nrow(gini), by = 2)],
    rep.median - dem.median)
## [1] 0.9418128
```

We find that the correlation is positive and quite high, indicating that political polarization and income inequality move in a similar direction. As we have already emphasized, this correlation alone does not imply causality. Many variables have an upwards trend during this time, leading to a high positive correlation among them.

3.6.3 QUANTILE–QUANTILE PLOT

Finally, in some cases, we are interested in comparing the entire distributions of two variables rather than just the mean or median. One way to conduct such a comparison is to simply plot two histograms side-by-side. As an example, we compare the distribution of ideal points on the racial liberalism/conservatism dimension in the 112th Congress. When comparing across multiple plots, it is important to use the same scales for the horizontal and vertical axes for all plots to facilitate the comparison.

```
hist(dem112$dwnom2, freq = FALSE, main = "Democrats",
     xlim = c(-1.5, 1.5), ylim = c(0, 1.75),
     xlab = "Racial liberalism/conservatism dimension")
hist(rep112$dwnom2, freq = FALSE, main = "Republicans",
     xlim = c(-1.5, 1.5), ylim = c(0, 1.75),
     xlab = "Racial liberalism/conservatism dimension")
```

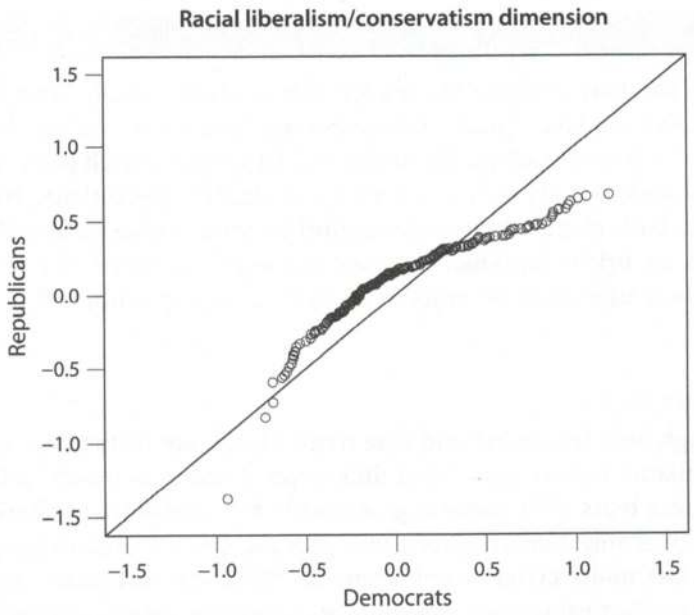


We observe that the two distributions are similar, though the distribution for Democrats appears to have a longer upper tail (i.e., the distribution extends further to the right) than that for Republicans. In addition, the Republicans' ideological positions seem to have a greater concentration towards the center than those of the Democrats.

A more direct way of comparing two distributions is a *quantile–quantile plot* or *Q–Q plot*. The Q–Q plot is based on *quantiles*, defined in section 2.6.1. It is a scatter plot of quantiles where each point represents the same quantile. For example, the median, upper quartile, and lower quartile of one sample will be plotted against the corresponding quantiles of the other sample. If two distributions are identical, then all quantiles have the same values. In this case, the Q–Q plot will result in the 45-degree line. Points above the 45-degree line indicate that a variable plotted on the vertical axis has a greater value at the corresponding quantile than a variable on the horizontal axis. In contrast, points below a 45-degree line imply the opposite relationship. This implies, for example, that if all points are above the 45-degree line, the variable on the vertical axis takes a greater value in every quantile than the variable on the horizontal axis.

Another useful feature of the Q–Q plot is that we can check the relative dispersion of two distributions. If the points in a Q–Q plot form a flatter line than the 45-degree line, they indicate that the distribution plotted on the horizontal axis is more dispersed than that on the vertical axis. In contrast, if the line has a steeper slope than 45 degrees, then the distribution plotted on the vertical line has a greater spread. The `qqplot()` function generates this plot by specifying the arguments `x` and `y`.

```
qqplot(dem112$dwnom2, rep112$dwnom2, xlab = "Democrats",
       ylab = "Republicans", xlim = c(-1.5, 1.5), ylim = c(-1.5, 1.5),
       main = "Racial liberalism/conservatism dimension")
abline(0, 1) # 45-degree line
```



In this Q–Q plot, the horizontal and vertical axes represent the racial dimension for Democrats and Republicans, respectively. The fact that the points representing lower quantiles appear above the 45-degree line indicate that liberal Republicans are more conservative than liberal Democrats. This is because these quantiles have greater values (i.e., more conservative) for Republicans than the corresponding quantiles for Democrats. In contrast, the points representing upper quantiles are located below the 45-degree line. That is, at the highest quantiles, i.e., the conservative ones, the Democrats score higher and so more conservatively than the Republicans. Thus, conservative Democrats are more conservative than conservative Republicans. Conservative Republicans would be more conservative than conservative Democrats if all the points for the upper quantiles were above the 45-degree line. Finally, the line connecting the points is flatter than the 45-degree line, indicating that the distribution of ideological positions is more dispersed for Democrats than for Republicans.

The **quantile–quantile plot** or Q–Q plot is a scatter plot of quantiles. It plots the value of each quantile for one variable against the value of the corresponding quantile for another variable. If the distributions of the two variables are identical, all points of the Q–Q plot lie on the 45-degree line. If the points form a line whose slope is steeper than 45 degrees, the distribution plotted on the vertical axis is more dispersed than the distribution on the horizontal axis. If the slope is less than 45 degrees, then the distribution on the vertical axis has less dispersion.

3.7 Clustering

In the previous analysis, the scatter plot made it visually clear that the 112th Congress had two ideologically distinct groups, Democrats and Republicans. But, are there any *clusters* of ideologically similar legislators within each party? Is there a well-defined procedure that can uncover groups of similar observations? We consider one of the most basic *clustering algorithms*, called *k-means*. Before we describe the *k*-means algorithm, we briefly introduce two new important R objects: *matrix* and *list*. These objects will be used when we implement the *k*-means algorithm in R.

3.7.1 MATRIX IN R

Although both the matrix and data frame objects are rectangular arrays and have many similarities, there are critical differences. Most importantly, a data frame can take different types of variables (e.g., numeric, factor, character) whereas a matrix in principle takes only numeric values (though it also can accommodate logical and other special values under certain circumstances). While one can extract variables from a data frame object using the `$` operator, in general the entries of a matrix need to be extracted by using square brackets `[,]` whose first and second elements, separated by a comma, indicate the rows and columns of interest, respectively. Although we do not exploit it in this book, a matrix is useful for *linear algebra* operations and is generally more computationally efficient than a data frame.

To create a matrix object, we can use the `matrix()` function by specifying the size of the matrix via the `nrow` (number of rows) and `ncol` (number of columns) arguments and indicating whether the matrix should be filled with the input data by row (`byrow = TRUE`) or column (`byrow = FALSE`). Moreover, adding labels to rows and columns can be done by the `rownames()` and `colnames()` functions.

```
## 3x4 matrix filled by row; first argument takes actual entries
x <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
rownames(x) <- c("a", "b", "c")
colnames(x) <- c("d", "e", "f", "g")
dim(x) # dimension
## [1] 3 4

x
##   d  e  f  g
## a 1  2  3  4
## b 5  6  7  8
## c 9 10 11 12
```

If one coerces a data frame object into a matrix using the `as.matrix()` function, some features of the data frame object, such as variable types, will get lost. In

the following example, we illustrate the fact that a data frame can take different data types such as character and numeric, but a matrix cannot accommodate them. Instead, the `as.matrix()` function converts variables of different types to a single type, `character` in this case.

```
## data frame can take different data types
y <- data.frame(y1 = as.factor(c("a", "b", "c")), y2 = c(0.1, 0.2, 0.3))
class(y$y1)

## [1] "factor"

class(y$y2)

## [1] "numeric"

## as.matrix() converts both variables to character
z <- as.matrix(y)
z

##      y1  y2
## [1,] "a" "0.1"
## [2,] "b" "0.2"
## [3,] "c" "0.3"
```

Finally, some useful operations on a matrix include `colSums()` and `colMeans()`, which calculate the column sums and means, respectively. The same operations can be applied to rows via the `rowSums()` and `rowMeans()` functions.

```
## column sums
colSums(x)

##  d  e  f  g
## 15 18 21 24

## row means
rowMeans(x)

##    a    b    c
##  2.5  6.5 10.5
```

More generally, we can use the `apply()` function to apply any function to a margin, meaning a row or a column, of a matrix. This function takes three main arguments: the first or `X` argument is a matrix, the second or `MARGIN` argument specifies a dimension over which we wish to apply a function (`1` represents rows while `2` represents columns), and the third or `FUN` argument names a function. We provide three examples. The first two examples are equivalent to the `colSums()` and `rowMeans()` shown above. The last example computes the standard deviation of each row.

```
## column sums
apply(x, 2, sum)
##  d  e  f  g
## 15 18 21 24

## row means
apply(x, 1, mean)
##   a   b   c
## 2.5 6.5 10.5

## standard deviation for each row
apply(x, 1, sd)
##        a        b        c
## 1.290994 1.290994 1.290994
```

3.7.2 LIST IN R

We now turn to another important object class in R, called a list. The list object is useful because it can store different types of objects as its elements. For example, a list can take numeric and character vectors of different lengths. In contrast, a data frame assumes those vectors to be of the same length. In fact, a list can even contain multiple data frames of different sizes as its elements. Therefore, a list is a very general class of objects.

Each element of a list comes with a name and can be extracted using the `$` operator (just like a variable in a data frame). It is also possible to extract an element using double square brackets, `[[ ]]`, with an integer or its element name indicating the element to be extracted. Below is a simple illustrative example of a list, which contains an integer vector of length 10 (`y1`), a character vector of length 3 (`y2`), and a data frame with two variables and three observations (`y3`). To create a list, we use the `list()` function and specify its elements by using their names as arguments.

```
## create a list
x <- list(y1 = 1:10, y2 = c("hi", "hello", "hey"),
          y3 = data.frame(z1 = 1:3, z2 = c("good", "bad", "ugly")))
## three ways of extracting elements from a list
x$y1 # first element
##  [1]  1  2  3  4  5  6  7  8  9 10

x[[2]] # second element
## [1] "hi"    "hello" "hey"
```

```
x[["y3"]] # third element
##   z1   z2
## 1  1 good
## 2  2  bad
## 3  3 ugly
```

Some of the functions we introduced can be applied to the list object. They include the `names()` (to extract the names of elements) and `length()` (to obtain the number of elements) functions.

```
names(x) # names of all elements
## [1] "y1" "y2" "y3"

length(x) # number of elements
## [1] 3
```

3.7.3 THE *k*-MEANS ALGORITHM

Now that we are familiar with matrices and lists, we can use them to apply the *k*-means algorithm. The *k*-means algorithm is an *iterative algorithm* in which a set of operations are repeatedly performed until a noticeable difference in results is no longer produced. The goal of the algorithm is to split the data into k similar groups where each group is associated with its *centroid*, which is equal to the within-group mean. This is done by first assigning each observation to its closest cluster and then computing the centroid of each cluster based on this new cluster assignment. These two steps are iterated until the cluster assignment no longer changes. The algorithm is defined as follows.

The ***k*-means algorithm** produces the prespecified number of clusters k and consists of the following steps:

Step 1: Choose the initial centroids of k clusters.

Step 2: Given the centroids, assign each observation to a cluster whose centroid is the closest (in terms of Euclidean distance) to that observation.

Step 3: Choose the new centroid of each cluster whose coordinate equals the within-cluster mean of the corresponding variable.

Step 4: Repeat Step 2 and 3 until cluster assignments no longer change.

Note that the researchers must choose the number of clusters k and the initial centroid of each cluster. In R, the initial locations of centroids are randomly selected, unless otherwise specified.

It is typically a good idea to *standardize* the inputs before applying the k-means algorithm. Doing so brings all variables to the same scale so that the clustering result does not depend on how each variable is measured. This is done by computing the *z-score* introduced earlier (see equation (3.1)). Recall that we compute the z-score of a variable by subtracting the mean from it (called *centering*) and then dividing it by the standard deviation (called *scaling*). In R, we can standardize a variable or a set of variables using the `scale()` function, which takes either a vector of a single variable or a matrix of multiple variables.

Going back to our study of partisanship, we apply the k-means clustering algorithm separately to the DW-NOMINATE scores for the 80th and 112th Congresses. We choose $k = 2$ and $k = 4$, producing 2 and 4 clusters, respectively. The function `kmeans()` implements the k-means algorithm in R. The function has various arguments, but the first argument `x` takes a *matrix* of observations to which one applies the k-means algorithm. For our application, this matrix has two columns, representing the first and second dimensions of DW-NOMINATE scores, and the number of rows equals the number of legislators in each Congress. We use the `cbind()` (or "column bind") function to combine two variables by columns in order to create this matrix. As a side note, the `rbind()` (or "row bind") function allows one to bind two vectors or matrices by rows. We do not standardize the input variables in this application since the DW-NOMINATE scores are already scaled in a substantively meaningful manner.

```
dwnom80 <- cbind(congress$dwnom1[congress$congress == 80],
                 congress$dwnom2[congress$congress == 80])
dwnom112 <- cbind(congress$dwnom1[congress$congress == 112],
                  congress$dwnom2[congress$congress == 112])
```

The main arguments of the `kmeans()` function include `centers` (the number of clusters), `iter.max` (the maximum number of iterations), and `nstart` (the number of randomly chosen initial centroids). It is recommended that the `nstart` argument is specified so that the algorithm is run several times with different starting values (the `kmeans()` function reports the best results). We begin by fitting the k-means algorithm with two clusters and five randomly selected starting values.

```
## k-means with 2 clusters
k80two.out <- kmeans(dwnom80, centers = 2, nstart = 5)
k112two.out <- kmeans(dwnom112, centers = 2, nstart = 5)
```

The output objects, `k80two.out` and `k112two.out`, are *lists*, which contain various elements regarding the results of the application of the k-means algorithm. They include `iter` (an integer representing the number of iterations until convergence, which is achieved when the cluster assignments no longer change), `cluster` (a vector of the resulting cluster membership), and `centers` (a matrix of cluster centroids).

```
## elements of a list
names(k80two.out)

## [1] "cluster"      "centers"      "totss"
```

```
## [4] "withinss"     "tot.withinss" "betweenss"
## [7] "size"         "iter"         "ifault"
```

As explained in section 3.7.2, the elements within each list can be accessed using $ like we access a variable in a data frame object. In both cases, the algorithm converged in just 1 iteration, which can be checked by examining the `iter` element of the output list object. The default maximum number of iterations is 10. If convergence is not achieved, the `iter.max` argument needs to be specified as a number greater than 10.

We now examine the final centroids of the resulting clusters using a 2-cluster model. Each output row shows a cluster with the horizontal and vertical coordinates of its centroid in the first and second columns, respectively.

```
## final centroids
k80two.out$centers
##          [,1]       [,2]
## 1  0.14681029 -0.3389293
## 2 -0.04843704  0.7827259

k112two.out$centers
##          [,1]       [,2]
## 1 -0.3912687 0.03260696
## 2  0.6776736 0.09061157
```

We next compute the numbers of Democratic and Republican legislators who belong to each cluster by creating a cross tabulation of party and cluster label variables.

```
## number of observations for each cluster by party
table(party = congress$party[congress$congress == 80],
      cluster = k80two.out$cluster)
##             cluster
## party          1   2
##   Democrat    62 132
##   Other        2   0
##   Republican 247   3

table(party = congress$party[congress$congress == 112],
      cluster = k112two.out$cluster)
##             cluster
## party          1   2
##   Democrat   200   0
##   Other        0   0
##   Republican   1 242
```

We find that for the 112th Congress, the k-means algorithm with 2 clusters produces one cluster containing all Democrats except one and the other consisting only of Republicans. While we chose the number of clusters to be 2 in this case, the algorithm discovers that these 2 clusters perfectly align on partisanship. In contrast, for the 80th Congress, one of the clusters contains a significant number of Democrats as well as Republicans. This is consistent with the fact that political polarization has worsened over time.

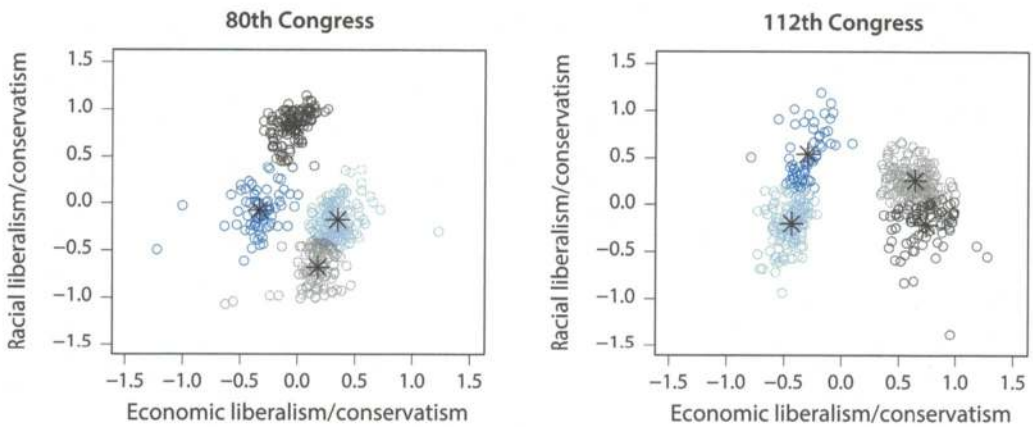
Next, we apply the k-means algorithm with 4 clusters and visualize the results. We begin by fitting the 4-cluster model to the 80th and 112th Congresses.

```
## k-means with 4 clusters
k80four.out <- kmeans(dwnom80, centers = 4, nstart = 5)
k112four.out <- kmeans(dwnom112, centers = 4, nstart = 5)
```

To visualize the results, we use the `plot()` function to create a scatter plot. The following syntax assigns different colors to observations that belong to different clusters. The centroid of each cluster is indicated by an asterisk.

```
## plotting the results using the labels and limits defined earlier
plot(dwnom80, col = k80four.out$cluster + 1, xlab = xlab, ylab = ylab,
     xlim = lim, ylim = lim, main = "80th Congress")
## plotting the centroids
points(k80four.out$centers, pch = 8, cex = 2)
## 112th Congress
plot(dwnom112, col = k112four.out$cluster + 1, xlab = xlab, ylab = ylab,
     xlim = lim, ylim = lim, main = "112th Congress")
points(k112four.out$centers, pch = 8, cex = 2)
```

For the full-color version of the plots, see page C2.



The `cex` argument given in the `points()` function controls the font size so that the centroid of each cluster is clearly visible. In addition, the `pch` argument specifies a certain symbol for plotting. Finally, we specify a vector of integer values, rather than actual color names, for the `col` argument so that each integer value is used for the corresponding cluster. We add 1 to the cluster labels so that we do not use black, the color of the cluster centroids, for the observations belonging to one of the clusters. The `palette()` function displays the exact correspondence between the color names and integer values (see section 5.3.3 for more details on the use of color in R).

```
palette()
## [1] "black"   "red"     "green3"  "blue"    "cyan"
## [6] "magenta" "yellow"  "gray"
```

The results show that the 4-cluster model splits the Democrats into 2 clusters and the Republicans into 2 clusters. Within each party, the division between the 2 clusters is clearest among the Democrats in the 80th Congress. For both parties, the within-party division is along the racial dimension. In contrast, the economic dimension dominates the difference between the two parties.

Clustering algorithms such as the k-means algorithm represent examples of *unsupervised learning* methods. Unlike in *supervised learning*, there is no outcome variable. Instead, the goal of unsupervised learning is to discover the hidden structures in data. The difficulty of unsupervised learning is that there is no clear measure of success and failure. In the absence of outcome data, it is difficult to know whether these clustering algorithms are producing the "correct" results. For this reason, human judgment is often required to make sure that the findings produced by clustering algorithms are reasonable.

3.8 Summary

This chapter focused on the issue of measurement. We discussed **survey sampling** as a principled and efficient way to infer the characteristics of a potentially large population from a small number of randomly sampled units without enumerating all units in the population. In chapter 2, we learned about the randomization of treatment assignment, which ensures that the treatment and control groups are equal on average in all aspects but the receipt of treatment. In survey sampling, we used the random sampling of units to make the sample representative of a target population. This allows researchers to infer population characteristics from the sample obtained from random sampling.

While random sampling is an effective technique, there are several complications in practice. First, while random sampling requires a complete list of potential units to be sampled, it is often difficult to obtain such a sampling frame. Second, due to cost and logistical constraints, researchers are forced to use complex random sampling techniques. Third, surveys typically lead to both **unit and item nonresponses**, which, if occurring nonrandomly, threaten the validity of inference. In recent years, the

nonresponse rate of phone surveys has dramatically increased. As a result, many polling firms are starting to use cheap Internet surveys through platforms like Qualtrics, even though many such surveys are not based on probability sampling. Beyond nonresponse problems, sensitive questions in surveys often result in **social desirability bias** in which respondents may falsify their answers and provide socially acceptable answers.

Furthermore, social scientists often face the question of how to measure latent concepts such as ideology and ability. We discussed an application of item response theory to political polarization in the US Congress. The idea is to infer legislators' ideological positions from their roll call votes. The same method was also applied to measure students' abilities from standardized tests. Using the estimated ideal points as an example, we also learned how to apply a basic **clustering algorithm** called the k-means algorithm in order to discover latent groups of observations with similar characteristics in data.

In addition to these concepts and methods, the chapter also introduced various numerical and visual summaries of data. While a **bar plot** summarizes the distribution of a factor variable, **box plots** and **histograms** are useful tools for depicting the distribution of continuous variables. The **correlation coefficient** numerically characterizes the association between two variables, whereas a **scatter plot** plots one variable against the other. Finally, unlike scatter plots, **quantile–quantile plots** (Q–Q plots) enable comparison of the distributions of two variables even when they are not measured in the same units.

3.9 Exercises

3.9.1 CHANGING MINDS ON GAY MARRIAGE: REVISITED

In this exercise, we revisit the gay marriage study we analyzed in section 2.8.2. It is important to work on that exercise before answering the following questions. In May 2015, three scholars reported several irregularities in the data set used to produce the results in the study.[3] They found that the gay marriage experimental data were statistically indistinguishable from data in the Cooperative Campaign Analysis Project (CCAP), which interviewed voters throughout the 2012 US presidential campaign. The scholars suggested that the CCAP survey data—and not the original data alleged to have been collected in the experiment—were used to produce the results reported in the gay marriage study. The release of a report on these irregularities ultimately led to the retraction of the original article. In this exercise, we will use several measurement strategies to reproduce the irregularities observed in the gay marriage data set.

To do so, we will use two CSV data files: a reshaped version of the original data set in which every observation corresponds to a unique respondent, `gayreshaped.csv` (see table 3.5), and the 2012 CCAP data set alleged to have been used as the basis for the gay marriage study results, `ccap2012.csv` (see table 3.6). Note that the feeling

[3] This exercise is based on the unpublished report "Irregularities in LaCour (2014)" by David Broockman, Joshua Kalla, and Peter Aronow.

Table 3.5. Gay Marriage Reshaped Data.

Variable	*Description*
`study`	which study the data set is from (`1` = study 1, `2` = study 2)
`treatment`	five possible treatment assignment options
`therm1`	survey thermometer rating of feeling towards gay couples in wave 1 (`0-100`)
`therm2`	survey thermometer rating of feeling towards gay couples in wave 2 (`0-100`)
`therm3`	survey thermometer rating of feeling towards gay couples in wave 3 (`0-100`)
`therm4`	survey thermometer rating of feeling towards gay couples in wave 4 (`0-100`)

Note: See table 2.7 for the original data.

Table 3.6. 2012 Cooperative Campaign Analysis Project (CCAP) Survey Data.

Variable	*Description*
`caseid`	unique respondent ID
`gaytherm`	survey thermometer rating of feeling towards gay couples (`0-100`)

thermometer measures how warmly respondents feel towards gay couples on a 0–100 scale.

1. In the gay marriage study, researchers used seven waves of a survey to assess how lasting the persuasion effects were over time. One irregularity the scholars found is that responses across survey waves in the control group (where no canvassing occurred) had unusually high correlation over time. What is the correlation between respondents' feeling thermometer ratings in waves 1 and 2 for the control group in study 1? To handle missing data, we should set the `use` argument of the `cor()` function to `"complete.obs"` so that the correlation is computed using only observations that have no missing data. Provide a brief substantive interpretation of the results.

2. Repeat the previous question using study 2 and comparing all waves within the control group. Note that the `cor()` function can take a single data frame with multiple variables. To handle missing data in this case, we can set the `use` argument to `"pairwise.complete.obs"`. This means that the `cor()` function uses all observations that have no missing values for a given pair of waves even if some of them have missing values in other waves. Briefly interpret the results.

3. Most surveys find at least some *outliers* or individuals whose responses are substantially different from the rest of the data. In addition, some respondents may change their responses erratically over time. Create a scatter plot to visualize the relationships between wave 1 and each of the subsequent waves in study 2. Use only the control group. Interpret the results.

4. The researchers found that the data of the gay marriage study appeared unusually similar to the 2012 CCAP data set even though they were supposed to be samples of completely different respondents. We use the data contained in `ccap2012.csv` and `gayreshaped.csv` to compare the two samples. Create a histogram of the 2012 CCAP feeling thermometer, the wave-1 feeling thermometer from study 1, and the wave-1 feeling thermometer from study 2. There are a large number of missing values in the CCAP data. Consider how the missing data might have been recoded in the gay marriage study. To facilitate the comparison across histograms, use the `breaks` argument in the `hist()` function to keep the bin sizes equal across histograms. Briefly comment on the results.

5. A more direct way to compare the distributions of two samples is through a *quantile–quantile plot*. Use this visualization method to conduct the same comparison as in the previous question. Briefly interpret the plots.

3.9.2 POLITICAL EFFICACY IN CHINA AND MEXICO

In 2002, the World Health Organization conducted a survey of two provinces in China and three provinces in Mexico.[4] One issue of interest, which we analyze in this exercise, concerns political efficacy. First, the following self-assessment question was asked.

> How much say do you have in getting the government to address issues that interest you?
>
> (5) Unlimited say, (4) A lot of say, (3) Some say, (2) Little say, (1) No say at all.

After the self-assessment question, three vignette questions were asked.

> [Alison] lacks clean drinking water. She and her neighbors are supporting an opposition candidate in the forthcoming elections that has promised to address the issue. It appears that so many people in her area feel the same way that the opposition candidate will defeat the incumbent representative.
>
> [Jane] lacks clean drinking water because the government is pursuing an industrial development plan. In the campaign for an upcoming election, an opposition party has promised to address the issue, but she feels it would be futile to vote for the opposition since the government is certain to win.
>
> [Moses] lacks clean drinking water. He would like to change this, but he can't vote, and feels that no one in the government cares about this issue. So he suffers in silence, hoping something will be done in the future.

The respondent was asked to assess each vignette in the same manner as the self-assessment question.

[4] This exercise is based on Gary King, Christopher J.L. Murray, Joshua A. Salomon, and Ajay Tandon (2004) "Enhancing the validity and cross-cultural comparability of measurement in survey research." *American Political Science Review*, vol. 98, no. 1 (February), pp. 191–207.

Table 3.7. Vignette Survey Data.

Variable	*Description*
`self`	self-assessment response
`alison`	response to the Alison vignette
`jane`	response to the Jane vignette
`moses`	response to the Moses vignette
`china`	`1` for China and `0` for Mexico
`age`	age of respondent in years

How much say does ["name"] have in getting the government to address issues that interest [him/her]?

(5) Unlimited say, (4) A lot of say, (3) Some say, (2) Little say, (1) No say at all.

["name"] is replaced by either Alison, Jane, or Moses.

The data set we analyze `vignettes.csv` contains the variables whose names and descriptions are given in table 3.7. In the analysis that follows, we assume that these survey responses can be treated as numerical values. For example, "Unlimited say" = 5, and "Little say" = 2. This approach is not appropriate if, for example, the difference between "Unlimited say" and "A lot of say" is not the same as the difference between "Little say" and "No say at all." However, relaxing this assumption is beyond the scope of this chapter.

1. We begin by analyzing the self-assessment question. Plot the distribution of responses separately for China and Mexico using bar plots, where the vertical axis is the proportion of respondents. In addition, compute the mean response for each country. According to this analysis, which country appears to have a higher degree of political efficacy? How does this evidence match with the fact that in the 2000 election, Mexican citizens voted out of office the ruling Institutional Revolutionary Party (PRI) who had governed the country for more than 80 years, while Chinese citizens have not been able to vote in a fair election to date?

2. We examine the possibility that any difference in the levels of efficacy between Mexican and Chinese respondents is due to the difference in their age distributions. Create histograms for the age variable separately for Mexican and Chinese respondents. Add a vertical line representing the median age of the respondents for each country. In addition, use a quantile–quantile plot to compare the two age distributions. What differences in age distribution do you observe between the two countries? Answer this question by interpreting each plot.

3. One problem with the self-assessment question is that survey respondents may interpret the question differently. For example, two respondents who choose the same answer may be facing quite different political situations and hence may interpret "A lot of say" differently. To address this problem, we rank a respondent's answer to the self-assessment question relative to the same respondent's answer

to a vignette question. Compute the proportion of respondents, again separately for China and Mexico, who rank themselves (according to the self-assessment question) as having less say in the government's decisions than Moses (the last vignette). How does the result of this analysis differ from that of the previous analysis? Give a brief interpretation of the result.

4. We focus on survey respondents who ranked these three vignettes in the expected order (i.e., Alison $\geq$ Jane $\geq$ Moses). Create a variable that represents how respondents rank themselves relative to these vignettes. This variable should be equal to `1` if respondents rank themselves less than Moses, `2` if ranked the same as Moses or between Moses and Jane, `3` if ranked the same as Jane or between Jane and Alison, and `4` if ranked the same as Alison or higher. Create the bar plots of this new variable as done in question 1. The vertical axis should represent the proportion of respondents for each response category. Also, compute the mean value of this new variable separately for China and Mexico. Give a brief interpretation of the result by comparing these results with those obtained in question 1.

5. Is the problem identified above more or less severe among older respondents when compared to younger ones? Answer the previous question separately for those who are 40 years or older and those who are younger than 40 years. Does your conclusion for the previous question differ between these two groups of respondents? Relate your discussion to your finding for question 2.

3.9.3 VOTING IN THE UNITED NATIONS GENERAL ASSEMBLY

Like legislators in the US Congress, the member states of the United Nations (UN) are politically divided on many issues such as trade, nuclear disarmament, and human rights. During the Cold War, countries in the UN General Assembly tended to split into two factions: one led by the capitalist United States and the other by the communist Soviet Union. In this exercise, we will analyze how states' ideological positions, as captured by their votes on UN resolutions, have changed since the fall of communism.[5] Table 3.8 presents the names and descriptions of the variables in the data set contained in the CSV file `unvoting.csv`.

In the analysis that follows, we measure state preferences in two ways. First, we can use the proportion of votes by each country that coincide with votes on the same issue cast by the two major Cold War powers: the United States and the Soviet Union. For example, if a country voted for 10 resolutions in 1992, and if its vote matched the United States's vote on exactly 6 of these resolutions, the variable `PctAgreeUS` in 1992 would equal 60 for this country. Second, we can also measure state preferences in terms of numerical ideal points as explained in section 3.5. These ideal points capture what international relations scholars have called countries' *liberalism* on issues such as political freedom, democratization, and financial liberalization. The two measures

[5] This exercise is based on Michael A. Bailey, Anton Strezhnev, and Erik Voeten (2015) "Estimating dynamic state preferences from United Nations voting data." *Journal of Conflict Resolution*, doi = 10.1177/0022002715595700.

Table 3.8. United Nations Ideal Points Data.

Variable	*Description*
`CountryName`	name of the country
`CountryAbb`	abbreviated name of the country
`idealpoint`	its estimated ideal point
`Year`	year for which the ideal point is estimated
`PctAgreeUS`	proportion of votes that match with votes cast by the United States on the same issue
`PctAgreeRUSSIA`	proportion of votes that match with votes cast by Russia/the Soviet Union on the same issue

are highly correlated, with larger (more liberal) ideal points corresponding to a higher proportion of votes that agree with the United States.

1. We begin by examining how the distribution of state ideal points has changed since the end of communism. Plot the distribution of ideal points separately for 1980 and 2000—about 10 years before and 10 years after the fall of the Berlin Wall, respectively. Add the median to each plot as a vertical line. How do the two distributions differ? Pay attention to the degree of polarization and give a brief substantive interpretation of the results. Use the `quantile()` function to quantify the patterns you identified.

2. Next, examine how the number of countries voting with the United States has changed over time. Plot the average percentage agreement with the United States across all countries over time. Also, add the average percentage agreement with Russia as another line for comparison. Using the `tapply()` function may help with this analysis. Does the United States appear to be getting more or less isolated over time, as compared to Russia? Identify some countries that are consistently pro-US. What are the most pro-Russian countries? Give a brief substantive interpretation of the results.

3. One problem with using the proportion of votes that agree with the United States or Russia as a measure of state preferences is that the ideological positions, and consequently the voting patterns, of the two countries might themselves have changed over time. This makes it difficult to know which countries' ideological positions have changed. Investigate this issue by plotting the evolution of the two countries' ideal points over time. Add the yearly median ideal point of all countries. How might the results of this analysis modify (or not) your interpretation of the previous analysis?

4. Let's examine how countries that were formerly part of the Soviet Union differ in terms of their ideology and UN voting compared to countries that were not part of the Soviet Union. The former Soviet Union countries are Estonia, Latvia, Lithuania, Belarus, Moldova, Ukraine, Armenia, Azerbaijan, Georgia,

Kazakhstan, Kyrgyzstan, Tajikistan, Turkmenistan, Uzbekistan, and Russia. The `%in%` operator, which is used as `x %in% y`, may be useful. This operator returns a logical vector whose elements are `TRUE` if the corresponding element of vector `x` is equal to a value contained in vector `y` and otherwise `FALSE`. Focus on the most recently available UN data from 2012 and plot each post-Soviet Union state's ideal point against the proportion of its votes that agree with the United States. Compare the post-Soviet Union states, within the same plot, against the other countries. Briefly comment on what you observe.

5. We have just seen that while some post-Soviet countries have retained nonliberal ideologies, other post-Soviet countries were much more liberal in 2012. Let's examine how the median ideal points of Soviet/post-Soviet countries and all other countries have varied over all the years in the data. Plot these median ideal points by year. Be sure to indicate 1989, the year of the fall of the Berlin Wall, on the graph. Briefly comment on what you observe.

6. Following the end of communism, countries that were formerly part of the Soviet Union have become much more ideologically diverse. Is this also true of the world as a whole? In other words, do countries still divide into two ideological factions? Let's assess this question by applying the k-means clustering algorithm to ideal points and the proportion of votes agreeing with the United States. Initiate the algorithm with just two centroids and visualize the results separately for 1989 and 2012. Briefly comment on the results.

Chapter 4

Prediction

Prophecy is a good line of business, but it is full of risks.
—Mark Twain, *Following the Equator*

In this chapter, we discuss prediction. Prediction is another important goal of data analysis in quantitative social science research. Our first example concerns the prediction of election outcomes using public opinion polls. We also show how to predict outcomes of interest using a linear regression model, which is one of the most basic statistical models. While many social scientists see causal inference as the ultimate goal of scholarly inquiry, prediction is often the first step towards understanding complex causal relationships that underlie human behavior. Indeed, valid causal inference requires the accurate prediction of counterfactual outcomes. Later in the chapter we discuss the connections between prediction and causal inference.

4.1 Predicting Election Outcomes

The 2008 US presidential election was historic. For the first time in American history, an African-American candidate, Barack Obama, was elected. This election was also important for the statistics community because a number of pundits accurately predicted the election outcome.

The United States's unique *Electoral College* system makes predicting election outcomes challenging. A candidate is elected to office by winning an absolute majority of electoral votes. Each of the 538 electors casts a single electoral vote. As of 2016, 535 of these votes are allocated among 50 states, corresponding to the 435 members of the House of Representatives and the 100 members of the Senate. The remaining 3 votes are given to the District of Columbia. In most cases, the electors vote for the candidate who won the plurality of votes in the state they represent, leading to a "winner-take-all" system in these states. In fact, some states have criminal penalties for voting for the candidate who did not win the plurality of votes. A winning presidential candidate must obtain at least 270 electoral votes.

Figure 4.1 shows the map of Electoral College votes for the 2008 election. See page C2 for the full-color version. Obama won 365 electoral votes (blue states), whereas

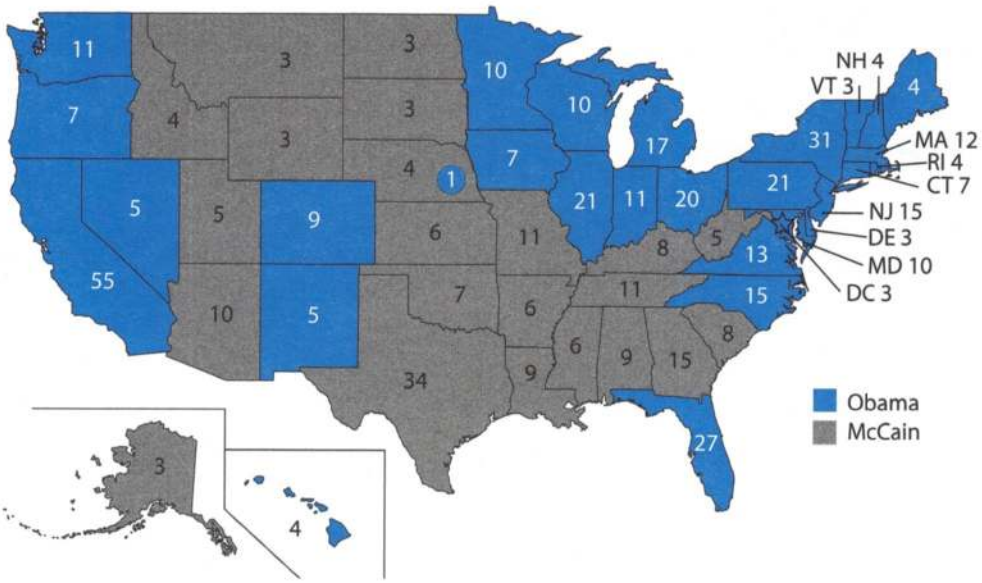


Figure 4.1. Electoral College Map of the 2008 US Presidential Election. The figure uses gray rather than red for the states won by McCain. See page C2 for the full-color version.

the Republican candidate John McCain received 173 votes (red states).[1] The Electoral College system implies that to successfully forecast the outcome of the US presidential election, we may need to accurately predict the winner of each state. Indeed, George W. Bush won the 2000 election by taking 25 electoral votes from Florida, where he defeated Al Gore by a slim margin of 537 votes after a controversial recount. As a result, Gore lost the election by the narrow margin of 5 electoral votes, even though he actually received a half million more popular votes than Bush at the national level. More recently, Donald Trump won the 2016 election even though Hillary Clinton received more votes nationally than Trump. Below, we show how to predict the election outcome using public opinion polls conducted within each state. Before we present the details of how this is done, we introduce two new programming concepts: loops and conditional statements.

4.1.1 LOOPS IN R

In many situations, we want to repeat the same operations multiple times where only small changes occur to the operations each time. For example, in order to forecast the result of the US presidential election, we must predict the election outcome within each state. This means that a similar set of computations will be performed a number of times. We would like to avoid writing nearly identical code chunks over and over again. A *loop* is a programming construct that allows us to repeatedly execute similar code chunks in a compact manner. The R syntax `for (i in X)` will create a loop, where `i` (or any other object name of your choice) is a *loop counter* that controls the

[1] Interestingly, Nebraska allocates two of its five electoral votes to the statewide winner while giving one electoral vote to the winner of each congressional district (Maine follows the same system). As a result, although McCain won a plurality of the popular vote in Nebraska, Obama received one electoral vote because he won the majority of votes in the second congressional district.

iterations of the loop, and X is the vector of values that the loop counter will successively take on. Consider the following pseudo-code.

```
for (i in X) {
    expression1
    expression2
    ...
    expressionN
}
```

Here, the collection of expressions from `expression1` through `expressionN` is repeated for each value `i` of the vector `X`. During each of these *iterations*, `i` takes on the corresponding value from the vector `X`, starting with the first element of `X` and ending with its last. Below is a simple example, which multiplies each number in a vector by 2. It is often useful to create an empty "container" vector whose elements are all `NA`s in order to store the results from computing all iterations. We use the `rep()` function to do this. Comments can be written into the loop as with any other code chunk in R. Braces { and } are used to denote the beginning and end of the body of the loop. When we start a loop (or related functions) in the RStudio text editor, the spacing will automatically indent and the closing bracket will align vertically with the `for` function. This makes the code easier to interpret and debug (i.e., identify and remove errors from the code).

```
values <- c(2, 4, 6)
n <- length(values) # number of elements in "values"
results <- rep(NA, n) # empty container vector for storing the results
## loop counter "i" will take values 1, 2, ..., n in that order
for (i in 1:n) {
    ## store the result of multiplication as the ith element of
    ## "results" vector
    results[i] <- values[i] * 2
    cat(values[i], "times 2 is equal to", results[i], "\n")
}
## 2 times 2 is equal to 4
## 4 times 2 is equal to 8
## 6 times 2 is equal to 12
results
## [1]  4  8 12
```

In each iteration of the above loop, the loop counter `i` takes an integer value, starting with `1` and ending with `n` with an increment of 1. Note that the `cat()` function, like `print()`, prints out an object on the screen. The `cat()` function combines multiple objects (character or other) into a character string, as inputs separated by commas. Without either the `cat()` or `print()` function, a loop will not print out

the `results[i]` value on the screen. Finally, recall that `\n` indicates the addition of a new line. Of course, in the above example, the loop is not strictly necessary because one can simply execute `values * 2`, which multiplies each element of the `values` vector by 2. Indeed, while loops may be conceptually easier, they are computationally intensive and so should be avoided whenever possible.

One important process is debugging code that involves a loop. Several strategies can reveal why a loop is not running properly. Since a loop simply executes the same command chunks many times, one could check whether the commands that go inside the loop can be executed without any error given a specific value of the loop counter. In the above example, one may simply try the following command before constructing the loop.

```
## check if the code runs when i = 1
i <- 1
x <- values[i] * 2
cat(values[i], "times 2 is equal to", x, "\n")

## 2 times 2 is equal to 4
```

Then, to make sure it behaves as we expect, we can change the first line to `i <- 2` or any other value that we want the loop counter `i` to take on. Another useful tip is to use the `print()` or `cat()` functions to print out the current value of the loop counter. This way, when there is an error, you always know how much of the loop succeeded. For example, if you cannot even run one iteration, there is likely something wrong with the code in the body of the loop. Alternatively, if the loop works for several iterations and then fails, perhaps something specific about the iteration that failed is causing the problem. The following example prints the iteration number to help identify the coding error. We use the `data.frame()` function to create an artificial data set with three variables, one of which is a character variable, and then attempt to compute the median of each variable using a loop.

```
## a toy data frame
data <- data.frame("a" = 1:2, "b" = c("hi", "hey"), "c" = 3:4)
## we see an error occurring at iteration 2
results <- rep(NA, 3)
for (i in 1:3) {
    cat("iteration", i, "\n")
    results[i] <- median(data[, i])
}

## iteration 1
## iteration 2

## Error in median.default(data[, i]): need numeric data

results

## [1]  1.5 NA NA
```

The loop was successfully executed in the first iteration but failed in the second iteration. This can be seen from the fact that an error message was printed before the printout of the `iteration 3` message. The reason for the failure is that the `median()` function takes numeric data only. As a result, the function produced an error in the second iteration, making the loop halt without computing the median for the second and third variables. This is indicated by `NA`s in the second and third elements of the `results` vector.

4.1.2 GENERAL CONDITIONAL STATEMENTS IN R

In section 2.2.4, we introduced simple conditional statements. We used the `ifelse()` function to create a vector of values where the elements of the resulting vector depend on an input object of the logical class. The general syntax is `ifelse(X, Y, Z)`. If an element `X` in the input is evaluated as `TRUE`, the value `Y` would be returned. If `X` is evaluated as `FALSE`, then the other value, `Z`, would be returned. This function is useful when recoding variables. Now, we will consider a more powerful form of conditional statements that can implement (or not) arbitrary chunks of R code depending on a logical expression. These take the form of `if(){}` and `if(){}else{}`. The first basic syntax is as follows.

```
if (X) {
    expression1
    expression2
    ...
    expressionN
}
```

If the value of `X` is `TRUE`, the code chunk `expression1` through `expressionN` will be executed. If the value of `X` is `FALSE`, then it will skip that code chunk entirely. The following simple example illustrates this.

```
## define the operation to be executed
operation <- "add"
if (operation == "add") {
    cat("I will perform addition 4 + 4\n")
    4 + 4
}
## I will perform addition 4 + 4
## [1] 8
if (operation == "multiply") {
    cat("I will perform multiplication 4 * 4\n")
    4 * 4
}
```

In the above code, the second portion of code on multiplication was not executed because the `operation` object was set to `"add"` rather than `"multiply"`. Thus, the expression `operation == "multiply"` returned a logical value of `FALSE`, indicating that the code chunk contained in the brackets is not performed. However, if `operation` is set to `"multiply"`, then `4 * 4`, rather than `4 + 4`, will be evaluated.

The `if(){}else{}` statements allow for greater flexibility by incorporating a set of R expressions to be evaluated if the argument in the `if()` function is `FALSE`. They contrast with the `if(){}` statements, which specify only the expressions to be evaluated when the argument in the `if()` function is `TRUE`. The following code will execute the code chunk `expression1a` through `expressionNa` if `X` is `TRUE` and the code chunk `expression1b` through `expressionNb` if `X` is `FALSE`.

```
if (X) {
    expression1a
    ...
    expressionNa
} else {
    expression1b
    ...
    expressionNb
}
```

Building on the earlier example, the following code illustrates how `if(){}else{}` statements work, implementing a different operation depending on the value of an object. Specifically, if the `operation` object is set to `"add"`, then the addition is performed, but otherwise, the multiplication is executed.

```
## note that "operation" is redefined
operation <- "multiply"
if (operation == "add") {
    cat("I will perform addition 4 + 4")
    4 + 4
} else {
    cat("I will perform multiplication 4 * 4")
    4 * 4
}

## I will perform multiplication 4 * 4
## [1] 16
```

One can construct even more complicated conditional statements using the `else if(){}` statement in the following manner.

```
if (X) {
    expression1a
    ...
    expressionNa
} else if (Y) {
    expression1b
    ...
    expressionNb
} else {
    expression1c
    ...
    expressionNc
}
```

The above syntax will execute the code chunk `expression1a` through `expressionNa` if condition `X` is met. If `X` is not met, but another condition `Y` is met, then the code chunk `expression1b` through `expressionNb` will be executed. Finally, if both `X` and `Y` are not satisfied, then the code chunk `expression1c` through `expressionNc` will be executed. Note that `else if()` can be repeated many times. In addition, the order of expressions matters. For example, if condition `Y` rather than `X` is evaluated first, then the code may produce a different result. Using `else if(){}`, we can modify the above example as follows.

```
## note that "operation" is redefined
operation <- "subtract"
if (operation == "add") {
    cat("I will perform addition 4 + 4\n")
    4 + 4
} else if (operation == "multiply") {
    cat("I will perform multiplication 4 * 4\n")
    4 * 4
} else {
    cat("'", operation, "' is invalid. Use either 'add' or 'multiply.'\n",
        sep = "")
}
## "subtract" is invalid. Use either "add" or "multiply."
```

Note that the `sep` argument specifies how each object should be separated. In the above example, `sep = ""` means that no character separates these objects. A separator can be any character string, commonly a comma and space (`sep = ", "`) or a semicolon and space (`sep = "; "`). The default is `sep = " "`, which will insert a space between objects.

Finally, conditional statements can be used effectively within a loop. Suppose, for example, that we want to perform a different arithmetic operation depending on whether an integer is even or odd. The following code first checks whether the input integer value is even or not. If it is even, R adds it to itself. If it is odd, R multiplies it. A message summarizing this operation is printed out for each iteration. In R, the `%%` operator computes the remainder of a division. For example, `5 %% 2` will return `1`, which is the remainder for the division of 5 by 2. If dividing an input integer value by 2 returns the remainder of 0 rather than 1, we conclude that it is an even number.

```
values <- 1:5
n <-length(values)
results <- rep(NA, n)
for (i in 1:n) {
    ## x and r get overwritten in each iteration
    x <- values[i]
    r <- x %% 2  # remainder when divided by 2 to check whether even or odd
    if (r == 0) { # remainder is zero
        cat(x, "is even and I will perform addition",
            x, "+", x, "\n")
        results[i] <- x + x
    } else { # remainder is not zero
        cat(x, "is odd and I will perform multiplication",
            x, "*", x, "\n")
        results[i] <- x * x
    }
}
## 1 is odd and I will perform multiplication 1 * 1
## 2 is even and I will perform addition 2 + 2
## 3 is odd and I will perform multiplication 3 * 3
## 4 is even and I will perform addition 4 + 4
## 5 is odd and I will perform multiplication 5 * 5
results
## [1]  1  4  9  8 25
```

Here, the code indentation, which is done automatically in RStudio, is important, making it clear that conditional statements are nested within a loop. The use of appropriate indentation is essential for writing computer code that contains loops and conditional statements.

4.1.3 POLL PREDICTIONS

Given that we now know how to use loops and conditional statements, we undertake the task of predicting the outcome of the 2008 US presidential election. Our forecast is based on a number of public opinion polls conducted before the election. The CSV data file `pres08.csv` contains the election results by state. In addition, we have the

Table 4.1. 2008 US Presidential Election Data.

Variable	*Description*
`state`	abbreviated name of the state
`state.name`	unabbreviated name of the state
`Obama`	Obama's vote share (percentage)
`McCain`	McCain's vote share (percentage)
`EV`	number of Electoral College votes for the state

Table 4.2. 2008 US Presidential Election Polling Data.

Variable	*Description*
`state`	abbreviated name of the state in which the poll was conducted
`Obama`	predicted support for Obama (percentage)
`McCain`	predicted support for McCain (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted

CSV file `polls08.csv`, which contains many polls within each state leading up to the election.[2] The names and descriptions of the variables in these data sets are given in tables 4.1 and 4.2, respectively. We begin by creating a variable, called `margin`, in both data frames, which represents Obama's vote margin over McCain in percentage points.

```
## load election results, by state
pres08 <- read.csv("pres08.csv")
## load polling data
polls08 <- read.csv("polls08.csv")
## compute Obama's margin
polls08$margin <- polls08$Obama - polls08$McCain
pres08$margin <- pres08$Obama - pres08$McCain
```

For each state, we generate a poll prediction for Obama's margin of victory using only the latest polls from the state. That is, we compute the mean prediction of all polls taken in the state on the day closest to the election. Note that this day may differ among states and there may be multiple polls conducted on the same day (more accurately, the same middate). To do this, we first initialize or create an empty vector of length 51, called `poll.pred`, which will contain the poll prediction for each of the 50 states and the District of Columbia. In the loop, we subset the data so that each iteration contains only the polls from one state.

[2] The polling data were obtained from http://electoral-vote.com.

We then further subset to extract the polls that were conducted within the state on the day closest to Election Day. This last step requires the conversion of the `middate` variable into the `Date` class using the `as.Date()` function. The `Date` class is useful because it can easily compute the number of days between the two specific dates. Input to the `as.Date()` function is a character string of the form `year-month-date` or `year/month/date`.

```
x <- as.Date("2008-11-04")
y <- as.Date("2008/9/1")
x - y # number of days between 2008/9/1 and 11/4

## Time difference of 64 days
```

Using this operation, we create the variable, called `DaysToElection`, which represents the number of days to the election. We compute this as the difference in days between the middate and Election Day (November 4). Finally, we compute the mean of poll predictions and store it as the corresponding element of `poll.pred`. Note that we use the `unique()` function to extract the unique state names in the code chunk below.

```
## convert to a Date object
polls08$middate <- as.Date(polls08$middate)
## compute the number of days to Election Day
polls08$DaysToElection <- as.Date("2008-11-04") - polls08$middate
poll.pred <- rep(NA, 51) # initialize a vector place holder
## extract unique state names which the loop will iterate through
st.names <- unique(polls08$state)
## add state names as labels for easy interpretation later on
names(poll.pred) <- as.character(st.names)
## loop across 50 states plus DC
for (i in 1:51){
    ## subset the ith state
    state.data <- subset(polls08, subset = (state == st.names[i]))
    ## further subset the latest polls within the state
    latest <- subset(state.data, DaysToElection == min(DaysToElection))
    ## compute the mean of latest polls and store it
    poll.pred[i] <- mean(latest$margin)
}
```

To set up the loop, we use the `unique()` function to extract the set of unique state names. Within the loop, we first subset the data for the `i`th state and store it as `state.data`. For example, if `i` equals `1`, it is Alabama and hence `st.names[i]` yields `AL`. We then further subset the data by extracting only the polls taken on the day closest to Election Day, which is indicated by the minimum value of the

`DaysToElection` variable. Finally, the resulting data `latest` is used to compute the average of the predicted margins from the latest polls.

We investigate the accuracy of our poll prediction by subtracting it from the actual election result of each state. The difference between the actual and predicted outcome is called the *prediction error*. We compute the prediction error by comparing the actual margin of victory with the predicted margin. We then compute the mean of poll prediction errors across states. This represents the average prediction error, which we call *bias*.

```
## error of latest polls
errors <- pres08$margin - poll.pred
names(errors) <- st.names # add state names
mean(errors) # mean prediction error

## [1] 1.062092
```

The result shows that on average across all states the poll predictions are approximately *unbiased*. More precisely, the mean of poll prediction errors across states is 1.1 percentage points, representing a bias of small magnitude. The poll predictions are for some states above and for other states below the actual election results, but on average these errors appear to roughly cancel out. While the poll predictions are approximately unbiased across states, the prediction for each state may not be accurate. For some states, the poll predictions may be well above the actual margins of victory, and these positive prediction errors are offset by large negative prediction errors for other states. To investigate this possibility, we compute the *root mean square* (RMS) of prediction error (see equation (2.3) introduced in section 2.6.2) or *root-mean-squared error* (RMSE), which represents the average magnitude of prediction error.

```
sqrt(mean(errors^2))

## [1] 5.90894
```

The result indicates that the average magnitude of each poll prediction error is about 6 percentage points.

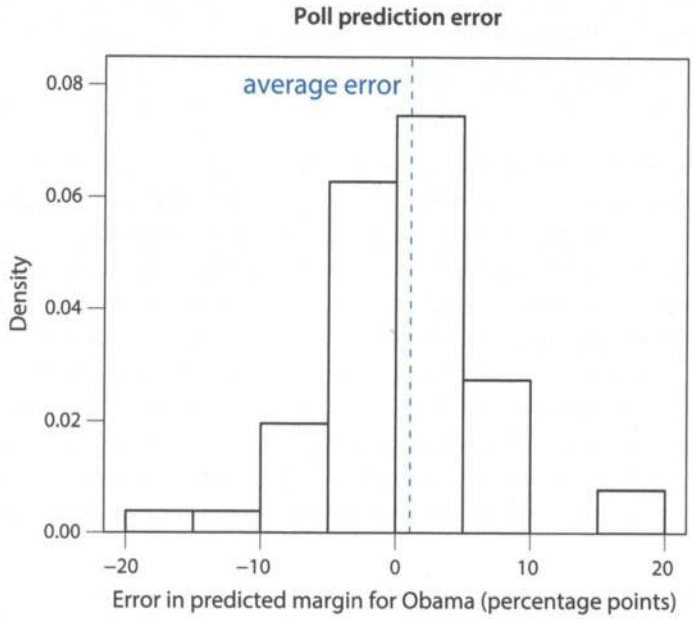
The **prediction error** is defined as

$$\text{prediction error} = \text{actual outcome} - \text{predicted outcome}.$$

The average prediction error is called **bias**, and prediction is said to be unbiased when its bias is zero. Finally, the root mean square of prediction error is called **root-mean-squared error**, representing the average magnitude of prediction error.

To obtain a more complete picture of prediction errors, we create a *histogram* using the `hist()` function (see section 3.3.2).

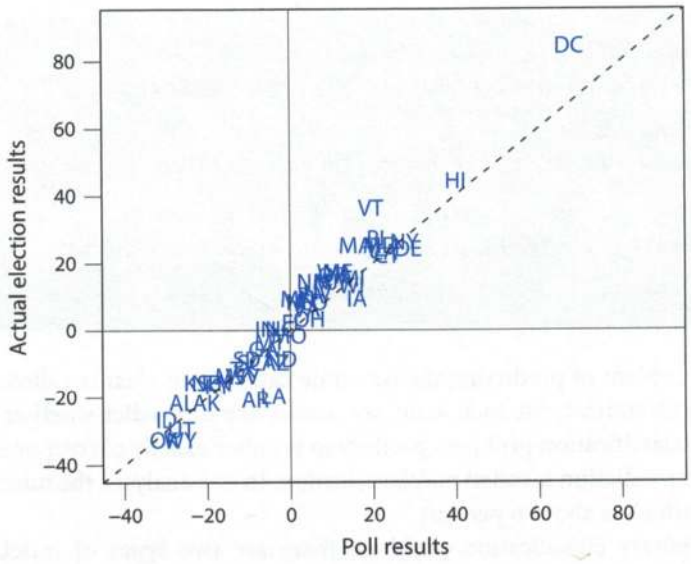
```
## histogram
hist(errors, freq = FALSE, ylim = c(0, 0.08),
     main = "Poll prediction error",
     xlab = "Error in predicted margin for Obama (percentage points)")
## add mean
abline(v = mean(errors), lty = "dashed", col = "blue")
text(x = -7, y = 0.08, "average error", col = "blue")
```



The histogram shows that the poll prediction error varies widely from one state to another. However, most errors are relatively small and larger errors are less likely to occur, yielding a *bell-shaped* distribution around zero.

We further examine the accuracy of poll predictions for each state by plotting them (horizontal axis) against the corresponding actual election results (vertical axis) using the two-letter state-name variable `state`. The states below (above) the 45-degree line indicate that the poll predictions were too favorable towards Obama (McCain). To plot text, we first create an "empty" plot by setting the `type` argument in the `plot()` function to `"n"` and then use the `text()` function to add state labels. As its first two arguments, the `text()` function takes the `x` and `y` coordinates for the location where the character string is to be plotted. The third argument of this function, `labels`, is a character vector of text labels to be plotted. In the current example, the x-coordinates and y-coordinates represent the poll predictions and Obama's actual margins.

```
## type = "n" generates "empty" plot
plot(poll.pred, pres08$margin, type = "n", main = "", xlab = "Poll results",
     xlim = c(-40, 90), ylim = c(-40, 90), ylab = "Actual election results")
## add state abbreviations
text(x = poll.pred, y = pres08$margin, labels = pres08$state, col = "blue")
## lines
abline(a = 0, b = 1, lty = "dashed") # 45-degree line
abline(v = 0)  # vertical line at 0
abline(h = 0)  # horizontal line at 0
```



Although for some states like the District of Columbia (DC) and Vermont (VT) the poll prediction is grossly inaccurate, this may not matter given that the US presidential election is essentially based on the winner-take-all system for each state. On the other hand, even when poll predictions are close to the actual election results in terms of percentage points, polls may predict the wrong candidate as the winner of a state. There are two types of prediction errors where the poll predictions chose the wrong winner. In the above plot, for the states that are plotted in the upper-left quadrant, Obama was predicted to lose (because the poll results are negative) but he actually won the states (because the actual election results are positive). Conversely, for the states in the lower-right quadrant, Obama was predicted to win but actually lost the states. The plot suggests that the poll predictions accurately chose the winner for most states. However, three states, which the poll predictions called wrongly, had a close race with the margin of victory approximately equal to 1 percentage point. We can use the `sign()` function to determine the sign of `poll.pred` and `pres08$margin` for each state. The function returns `1` if positive (Obama wins) and `-1` if negative (McCain wins) (`0` if zero, a tie).

Table 4.3. Confusion Matrix.

	Actual outcome: Positive	Actual outcome: Negative
Predicted outcome: Positive	true positive	false positive
Predicted outcome: Negative	false negative	true negative

Note: There are two types of correct classification, true positive and true negative. Similarly, false positive and false negative are two kinds of misclassification.

```
## which state polls called wrong?
pres08$state[sign(poll.pred) != sign(pres08$margin)]

## [1] IN MO NC
## 51 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI IA ID ... WY

## what was the actual margin for these states?
pres08$margin[sign(poll.pred) != sign(pres08$margin)]

## [1]  1 -1  1
```

The problem of predicting the outcome category or class is called *classification*. In the current context, for each state, we would like to predict whether Obama wins or not. In a classification problem, prediction is either exactly correct or incorrect, and an incorrect prediction is called *misclassification*. In our analysis, the misclassification rate is 3/51, which is about 6 percent.

In a binary classification problem, there are two types of misclassification. We may predict Obama to be the winner for a state where he actually lost the election. Conversely, Obama may be predicted to lose a state and yet in the actual election win it. If we regard Obama's victory (rather than his loss) as the "positive" outcome, then the former type of misclassification is called *false positive* whereas the latter is *false negative*. In the current example, Missouri (MO) is a false positive while Illinois (IN) and North Carolina (NC) are false negatives. Table 4.3 presents a *confusion matrix* where the two types of misclassification and correct classification are shown.

Classification refers to the problem of predicting a categorical outcome. Classification is either correct or incorrect. In a binary classification problem, there are two types of **misclassification**: false positive and false negative, representing incorrectly predicted positive and negative outcomes, respectively.

Finally, we can compute the number of Electoral College votes for Obama based on the poll predictions and compare it against the actual result, which was 364 votes. Since 270 votes was the winning threshold, the results show that the polls correctly

called Obama the elected president. The predicted total number of Electoral College votes was 15 fewer than the actual election result.[3]

```
## actual results: total number of electoral votes won by Obama
sum(pres08$EV[pres08$margin > 0])
## [1] 364
## poll prediction
sum(pres08$EV[poll.pred > 0])
## [1] 349
```

While the popular vote does not determine the election outcome, we can also examine the accuracy of national polls and how public opinion changed over the course of the campaign. To do this, we analyze the national polls contained in the CSV file `pollsUS08.csv`. The names and descriptions of the variables in this data set are identical to those of the last four variables in table 4.2. For each of the last 90 days of the campaign, we compute the average of support for each candidate using all polls taken within the past week and examine how it changes as Election Day nears. This can be done with a loop, where for a given day we take all polls that were conducted within the previous 7 days and on the corresponding day. We then compare these poll-based predictions against the actual vote shares in the election, which were 52.9% and 45.7% for Obama and McCain, respectively. Using the code for state polls above as a template, we construct the following code chunk.

```
## load the data
pollsUS08 <- read.csv("pollsUS08.csv")
## compute number of days to the election as before
pollsUS08$middate <- as.Date(pollsUS08$middate)
pollsUS08$DaysToElection <- as.Date("2008-11-04") - pollsUS08$middate
## empty vectors to store predictions
Obama.pred <- McCain.pred <- rep(NA, 90)
for (i in 1:90) {
    ## take all polls conducted within the past 7 days
    week.data <- subset(pollsUS08, subset = ((DaysToElection <= (90 - i + 7))
                                & (DaysToElection > (90 - i))))
    ## compute support for each candidate using the average
    Obama.pred[i] <- mean(week.data$Obama)
    McCain.pred[i] <- mean(week.data$McCain)
}
```

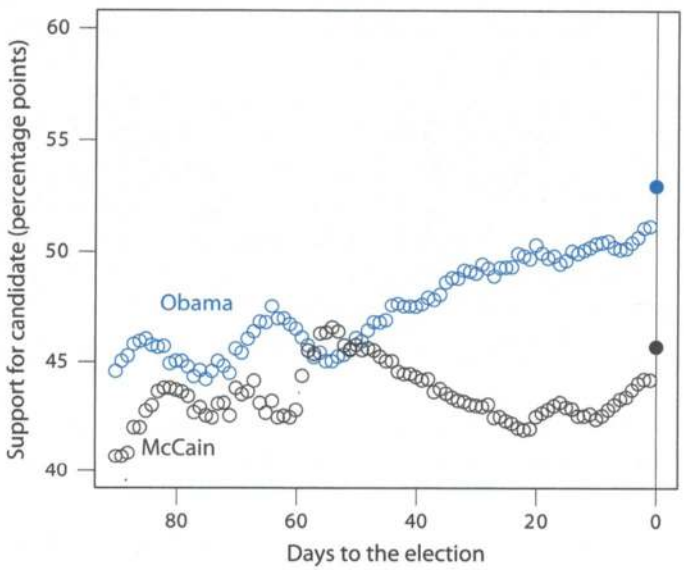
Note that in the above code we utilize shortcut syntax to assign the same value to multiple objects. Specifically, we use the single expression `x <- y <- z` rather than

[3] As noted earlier, Obama received one vote from Nebraska even though he lost the statewide vote.

two separate expressions, `x <- z` and `y <- z`, in order to assign the same value `z` to two objects, `x` and `y`. Furthermore, within the loop, we subset the data `pollsUS08` so that the resulting data contain only the polls conducted within the past 7 days and the day itself. For example, when the loop starts (i.e., `i` is equal to `1`), we subset the polls for which the `DaysToElection` variable is less than or equal to 96 ($= 90 - 1 + 7$) and greater than 89 ($= 90 - 1$). In the final iteration (i.e., `i` is equal to 90), this variable for the subsetted data takes a value less than or equal to 7 ($= 90 - 90 + 7$) and greater than 0 ($= 90 - 90$).

We now display the results using a *time-series plot*. We define the horizontal axis such that its leftmost value is 90 days prior to Election Day and its rightmost value is Election Day. This can be done by specifying the `xlim` argument to be `c(90, 0)` instead of `c(0, 90)`. The plot at the bottom uses gray circles rather than red for the states won by McCain. See page C3 for the full-color version.

```
## plot going from 90 days to 1 day before the election
plot(90:1, Obama.pred, type = "b", xlim = c(90, 0), ylim = c(40, 60),
     col = "blue", xlab = "Days to the election",
     ylab = "Support for candidate (percentage points)")
## type = "b" gives plot that includes both points and lines
lines(90:1, McCain.pred, type = "b", col = "red")
## actual election results: pch = 19 gives solid circles
points(0, 52.93, pch = 19, col = "blue")
points(0, 45.65, pch = 19, col = "red")
## line indicating Election Day
abline(v = 0)
## labeling candidates
text(80, 48, "Obama", col = "blue")
text(80, 41, "McCain", col = "red")
```



Which person is the most competent?

Figure 4.2. Example Pictures of Candidates Used in the Experiment. Source: A. Todorov et al. (2005) *Science*, vol. 308, no. 10 (June), pp. 1623–1626.

The resulting figure demonstrates the reasonable accuracy of preelection polls in terms of margin. Indeed, the Election Day margin (the difference between two solid circles) almost coincides with the predicted margin based on the polls taken within a week prior to the election. It is also interesting that public opinion shifts quite a bit during the course of campaign. Two months before the election, support for Obama was roughly tied with that for McCain. However, as Election Day approached, Obama's margin over McCain gradually increased. On Election Day, it was more than 7 percentage points. It is also worth noting that the proportion of other voters who were either undecided or supported third-party candidates declined.

4.2 Linear Regression

In the previous section, we used polling data to predict election outcomes. When doing so, we simply used the average of poll predictions. An alternative method of prediction is based on a statistical model. In this section, we introduce one of the most basic statistical models, called *linear regression*.

4.2.1 FACIAL APPEARANCE AND ELECTION OUTCOMES

Several psychologists have reported the intriguing result of an experiment showing that facial appearance predicts election outcomes better than chance.[4] In their experiment, the researchers briefly showed student subjects the black-and-white head shots of two candidates from a US congressional election (winner and runner-up). Figure 4.2 shows example pictures of the candidates from the 2004 Wisconsin Senate race. Russ Feingold of the Democratic Party (left) was the actual winner, and Tim Michels of the Republican Party (right) was the runner-up. The exposure of subjects to facial pictures lasted less than a second, and the subjects were then asked to evaluate the two candidates in terms of their perceived competence.

[4] This section is based on Alexander Todorov, Anesu N. Mandisodza, Amir Goren, and Crystal C. Hall (2005) "Inferences of competence from faces predict election outcomes." *Science*, vol. 308, no. 10 (June), pp. 1623–1626.

Table 4.4. Facial Appearance Experiment Data.

Variable	*Description*
`congress`	session of Congress
`year`	year of the election
`state`	state of the election
`winner`	name of the winner
`loser`	name of the runner-up
`w.party`	party of the winner
`l.party`	party of the loser
`d.votes`	number of votes for the Democratic candidate
`r.votes`	number of votes for the Republican candidate
`d.comp`	competence measure for the Democratic candidate
`r.comp`	competence measure for the Republican candidate

The researchers used these competence measures to predict election outcomes. Here, the competence measure for a Democratic candidate, for example, represents the proportion of experimental subjects who rated the Democrat more competent than the Republican. The key hypothesis is whether or not a within-a-second evaluation of facial appearance can predict election outcomes. The CSV data set, `face.csv`, contains the data from the experiment. Table 4.4 presents the names and descriptions of the variables in this data set. Note that we include data only from subjects who did not know the candidates' political parties, their policies, or even which candidate was the incumbent or challenger. They were simply making snap judgments about which candidate appeared more competent based on their facial expression alone.

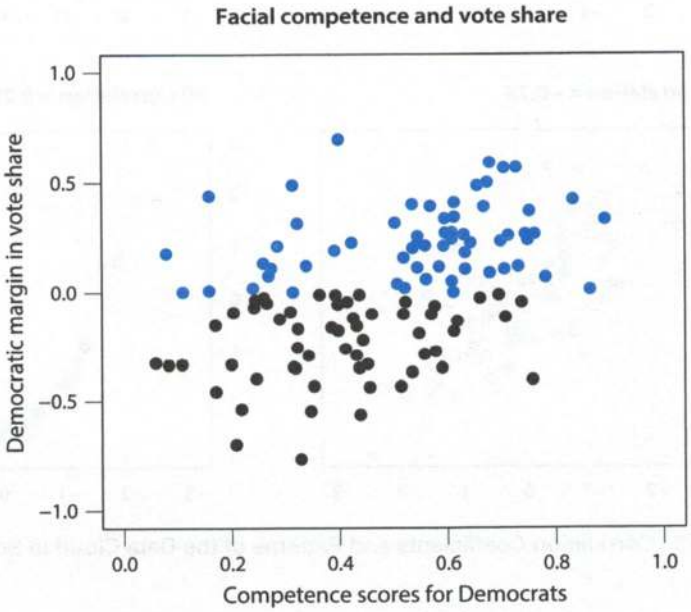
We begin our analysis of the facial appearance experiment data by creating a *scatter plot* of the competence measure against election outcomes. To do this, we create the win margins for Democratic candidates as the difference in two-party vote shares for Democratic and Republican candidates. Positive win margins favor Democrats. A two-party vote share is the number of votes each candidate receives out of just those votes cast for a major party candidate (not out of all votes cast).

```
## load the data
face <- read.csv("face.csv")
## two-party vote share for Democrats and Republicans
face$d.share <- face$d.votes / (face$d.votes + face$r.votes)
face$r.share <- face$r.votes / (face$d.votes + face$r.votes)
face$diff.share <- face$d.share - face$r.share
```

Next, we use the `plot()` function to generate a scatter plot. To make the symbols more informative, we can change them based on variables in our data set. The argument `pch` for the `plot()` function can specify the type of points to be plotted (see section 3.6). We use the `ifelse()` function when specifying the `col` argument so that red dots are used for the races with Republican winners and blue dots are used for those with Democratic winners. The plot shows a mild upward trend in the Democratic margin as the competence score for Democrats increases.

```
plot(face$d.comp, face$diff.share, pch = 16,
     col = ifelse(face$w.party == "R", "red", "blue"),
     xlim = c(0, 1), ylim = c(-1, 1),
     xlab = "Competence scores for Democrats",
     ylab = "Democratic margin in vote share",
     main = "Facial competence and vote share")
```

The plot below uses gray circles instead of red circles for Republican winners. See page C3 for the full-color version.



4.2.2 CORRELATION AND SCATTER PLOTS

We learned in section 3.6.2 that correlation represents the degree to which one variable is associated with another. A positive (negative) value of correlation means that one variable is more (less) likely to be above (below) its mean when the other variable is above its own mean. The upwards-sloping data cloud in the above scatter plot shows a positive *correlation* between perceived competence and vote share differential. To compute the correlation coefficient, we use the function `cor()`.

```
cor(face$d.comp, face$diff.share)
## [1] 0.4327743
```

This correlation of about 0.4 tells us that there is a moderately positive relationship between a candidate's perceived competence and his or her actual margin of victory on Election Day. That is, candidates who appear more competent than their

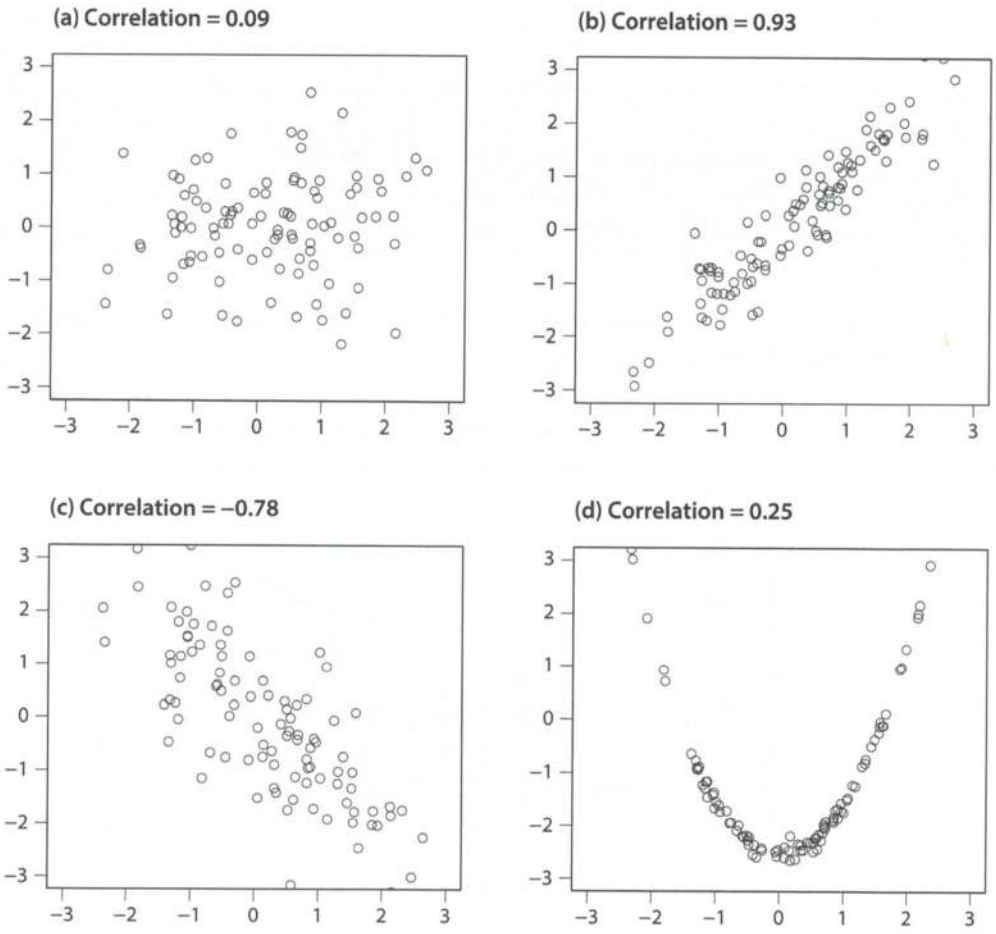


Figure 4.3. Correlation Coefficients and Patterns of the Data Cloud in Scatter Plots.

opponents—as rapidly judged by uninformed voters who don't recognize the candidates—are likely to win a higher share of the votes cast.

To get a better sense of the relationship between correlation coefficients and data cloud shapes, figure 4.3 presents four artificial data sets with various degrees of correlation. We observe that a positive (negative) correlation corresponds to an upwards (downwards) trend in the data cloud, and a greater magnitude of the correlation coefficient indicates a stronger linear relationship. Indeed, correlation represents a *linear relationship* between two variables. Perfect positive (negative) correlation, i.e., correlation of 1 (−1), would mean the two variables have a perfect linear relationship with data points located on a single line.

Thus, it is important for us to note that a lack of correlation does not necessarily imply a lack of a relationship. In panel (d), the correlation between the two variables is low but there is a clear *nonlinear relationship*, which in this case is a quadratic function.

The **correlation coefficient** quantifies the linear relationship between two variables. An upwards trend in the data cloud in a scatter plot implies a positive correlation, whereas a downwards trend in the data cloud represents a negative correlation. Correlation is often not suitable for representing a nonlinear relationship.

4.2.3 LEAST SQUARES

As shown above, correlation describes a linear relationship between two variables. However, such a relationship is best characterized using the following *linear model*:

$$Y = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} X + \underbrace{\epsilon}_{\text{error term}}. \tag{4.1}$$

In the model, Y is the outcome or response variable and X is the predictor or independent (explanatory) variable. In the current application, we will use the perceived competence measure as the predictor and the difference in two-party vote share as the outcome. Recall that any line can be defined by the *intercept* α and the *slope* parameter β. The intercept α represents the average value of Y when X is zero. The slope β measures the average increase in Y when X increases by one unit. The intercept and slope parameters are together called *coefficients*. The *error* (or *disturbance*) term, ϵ, allows an observation to deviate from a perfect linear relationship.

We use a model like this under the assumption that it approximates the *data-generating process* well. However, as well-known statistician George Box has stated, we must recognize that "all models are wrong, but some are useful." Even if the data are not generated according to the linear model specified in equation (4.1), the model can be a useful tool to predict the outcome of interest.

Since the values of α and β in equation (4.1) are unknown to researchers, they must be estimated from the data. In statistics, the estimates of parameters are indicated by "hats," where $\hat{\alpha}$ and $\hat{\beta}$ represent the estimates of α and β, respectively. Once we obtain the estimated values of coefficients α and β, then we have the so-called *regression line*. We can use this line to predict the value of the outcome variable given that of a predictor. Specifically, given a particular value of the predictor, $X = x$, we compute the *predicted value* (or *fitted value*) of the outcome variable, denoted by $\widehat{Y}$, using the regression function

$$\widehat{Y} = \hat{\alpha} + \hat{\beta}x. \tag{4.2}$$

Most likely, the predicted value will not equal the observed value. The difference between the observed outcome and its predicted value is called the *residual* or *prediction error*. Formally, we can write the residual as

$$\hat{\epsilon} = Y - \widehat{Y}. \tag{4.3}$$

Notice that the residual is represented by ϵ with a hat. Since the error term ϵ in equation (4.1) is unobserved, the residual represents an estimate of this error term.

The **linear regression model** is defined as

$$Y = \alpha + \beta X + \epsilon,$$

where Y is the outcome (or response) variable, X is the predictor or the independent (or explanatory) variable, ϵ is the error (or disturbance) term, and (α, β) are the coefficients. The slope parameter β represents the increase in the average outcome associated with a one-unit increase in the predictor. Once the estimates of the coefficients $(\hat{\alpha}, \hat{\beta})$ are obtained from the data, we can predict the outcome, using a given value of the predictor $X = x$, as $\widehat{Y} = \hat{\alpha} + \hat{\beta}x$. The difference between the observed outcome and this fitted or predicted value $\widehat{Y}$ is called the residual and is denoted by $\hat{\epsilon} = Y - \widehat{Y}$.

To fit a linear regression model in R, we use the `lm()` function. This function takes a formula of the form `Y ~ X` as the main argument where the outcome variable is `Y` and the predictor is `X`, taken from a data frame specified as the `data` argument. Note that an intercept will be automatically added to the regression model.

We now obtain the regression line for the facial appearance experiment data. We use the Democratic margin in the two-party vote share as the response variable and the perceived competence for Democratic candidates as the predictor.

```
fit <- lm(diff.share ~ d.comp, data = face) # fit the model
fit

##
## Call:
## lm(formula = diff.share ~ d.comp, data = face)
##
## Coefficients:
## (Intercept)       d.comp
##     -0.3122       0.6604
```

The output shows that the estimated intercept is -0.3122 whereas the estimated slope is 0.6604. That is, when no experimental subject thinks a Democratic candidate is more competent than a Republican counterpart, the predicted Democratic margin of two-party vote share is approximately -31.2 percentage points. If the perceived competence score increases by 10 percentage points, then the outcome variable is predicted to increase on average by 6.6 $(= 0.6604 \times 10)$ percentage points.

There is an alternative way of fitting the same model without the `data` argument. This requires specifying the entire names of objects for the outcome variable and the predictor as follows.

```
lm(face$diff.share ~ face$d.comp)
```

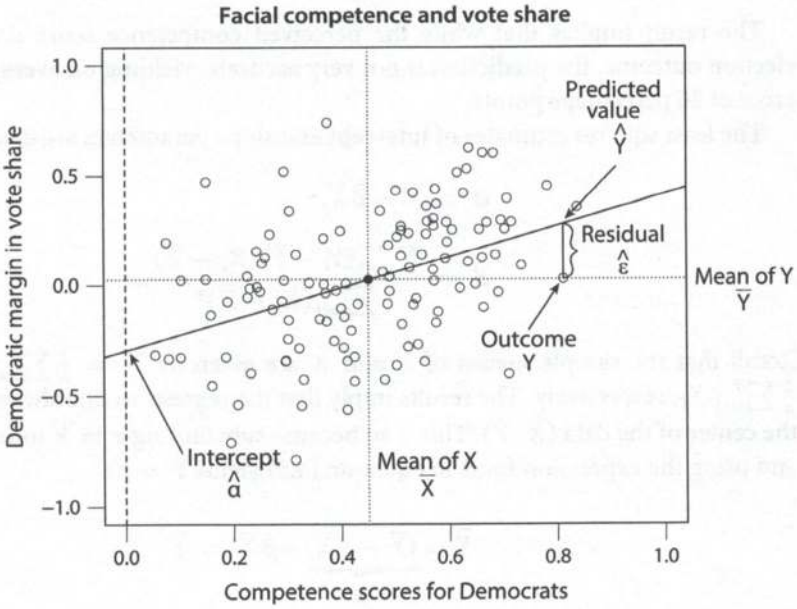
In general, this is not recommended because it unnecessarily complicates the syntax and may cause confusion. However, it may be useful when the variables we wish to use for regression exist as separate objects in the workspace.

In addition, to directly obtain the estimated coefficients ($\hat{\alpha}$, $\hat{\beta}$) and the predicted or fitted values $\widehat{Y}$, we can use the `coef()` and `fitted()` functions, respectively.

```
coef(fit)  # get estimated coefficients
## (Intercept)      d.comp
##  -0.3122259   0.6603815
head(fitted(fit))  # get fitted or predicted values
##           1           2           3           4           5
##  0.06060411 -0.08643340  0.09217061  0.04539236 0.13698690
##           6
##  -0.10057206
```

It is straightforward to add the regression line to the scatter plot using the `abline()` function which takes the output object from the `lm()` function as its input. The plot also shows the estimated intercept $\hat{\alpha}$ as well as the observed outcome Y, the predicted or fitted value $\hat{Y}$, and the residual $\hat{\epsilon}$ for one of the observations.

```
plot(face$d.comp, face$diff.share, xlim = c(0, 1.05), ylim = c(-1,1),
     xlab = "Competence scores for Democrats",
     ylab = "Democratic margin in vote share",
     main = "Facial competence and vote share")
abline(fit) # add regression line
abline(v = 0, lty = "dashed")
```



This regression line is the "line of best fit" because it minimizes the magnitude of prediction error. To estimate the line's intercept and slope parameters, a commonly used method is that of *least squares*. The idea is to choose $\hat{\alpha}$ and $\hat{\beta}$ such that together they minimize the *sum of squared residuals* (SSR), which is defined as

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta} X_i)^2. \tag{4.4}$$

In the equation, Y_i, X_i, and $\hat{\epsilon}_i$ represent the outcome variable, the predictor, and the residual, respectively, for the ith observation, and n is the sample size. The second and third equalities follow from the definition of the residual given in equations (4.3) and (4.2), respectively. The value of SSR is difficult to interpret. However, we can use the idea of *root mean square* (RMS) introduced in section 2.6.2 and applied earlier. Specifically, we can compute the *root-mean-squared error* (RMSE) as

$$\text{RMSE} = \sqrt{\frac{1}{n}\text{SSR}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \hat{\epsilon}_i^2}. \tag{4.5}$$

Therefore, RMSE represents the average magnitude of the prediction error for the regression, and this is what the method of least squares minimizes.

In R, RMSE can be easily calculated by first obtaining the residuals from the `resid()` function.

```
epsilon.hat <- resid(fit)   # residuals
sqrt(mean(epsilon.hat^2))   # RMSE

## [1] 0.2642361
```

The result implies that while the perceived competence score does predict the election outcome, the prediction is not very accurate, yielding on average a prediction error of 26 percentage points.

The least squares estimates of intercept and slope parameters are given by

$$\hat{\alpha} = \overline{Y} - \hat{\beta}\overline{X}, \tag{4.6}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})}{\sum_{i=1}^{n}(X_i - \overline{X})^2}. \tag{4.7}$$

Recall that the sample means of Y and X are given by $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, respectively. The results imply that the regression line always goes through the center of the data $(\overline{X}, \overline{Y})$. This is so because substituting $x = \overline{X}$ into equation (4.2) and using the expression for $\hat{\alpha}$ in equation (4.6) yields $\widehat{Y} = \overline{Y}$:

$$\widehat{Y} = \underbrace{(\overline{Y} - \hat{\beta}\overline{X})}_{\hat{\alpha}} + \hat{\beta}\overline{X} = \overline{Y}.$$

In the above plot, we observe that this is indeed the case. The regression line runs through the intersection of the vertical and horizontal dotted lines, which represent the means of X and Y, respectively.

In addition, when the method of least squares is used to estimate the coefficients, the predictions based on the fitted regression line are accurate on average. More precisely, the mean of residual $\hat{\epsilon}$ is zero, as the following algebraic manipulation shows:

$$\text{mean of } \hat{\epsilon} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\alpha} - \hat{\beta}X_i) = \overline{Y} - \hat{\alpha} - \hat{\beta}\overline{X} = 0.$$

In this equation, the first equality is due to the definition of the residual, the next equality is obtained by applying the summation for each term in the parentheses, and the final equality follows from equation (4.6). We emphasize that this is an algebraic equality and holds for *any* data set. In other words, a linear regression model always has zero average prediction error across all data points in the sample, but this does not necessarily mean that the linear regression model accurately represents the actual data-generating process.

A common method of estimating the coefficients of the linear regression model is the method of **least squares**, which minimizes the sum of squared residuals,

$$\text{SSR} = \textstyle\sum_{i=1}^{n}\hat{\epsilon}_i^2 = \sum_{i=1}^{n}(Y_i - \hat{\alpha} - \hat{\beta}X_i)^2.$$

The mean of residuals is always zero, and the regression line always goes through the center of data $(\overline{X}, \overline{Y})$ where $\overline{X}$ and $\overline{Y}$ are the sample means of X and Y, respectively.

It is also important to understand the relationship between the estimated slope of the regression and the correlation coefficient introduced in section 3.6.2:

$$\hat{\beta} = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \overline{Y})(X_i - \overline{X})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2}} \times \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2}}$$

$$= \text{correlation of } X \text{ and } Y \times \frac{\text{standard deviation of } Y}{\text{standard deviation of } X}. \tag{4.8}$$

The first equality holds because we divide and multiply the right hand side of equation (4.7) by the standard deviation of Y, i.e., $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$, whereas the second equality follows from the definitions of correlation and standard deviation (see equations (3.2) and (2.4), respectively).

The expression for the estimated slope parameter in equation (4.8) has two important implications. First, a positive (negative) correlation corresponds to a positive (negative) slope because standard deviations never take a negative value. Second, each increase of 1 standard deviation in X is associated with an average increase of ρ standard deviations in Y, where ρ is the correlation between X and Y. For example, if the correlation is 0.5, then a 1 standard deviation increase in X would result in a 0.5 standard deviation increase in Y. In the current example, the correlation between the

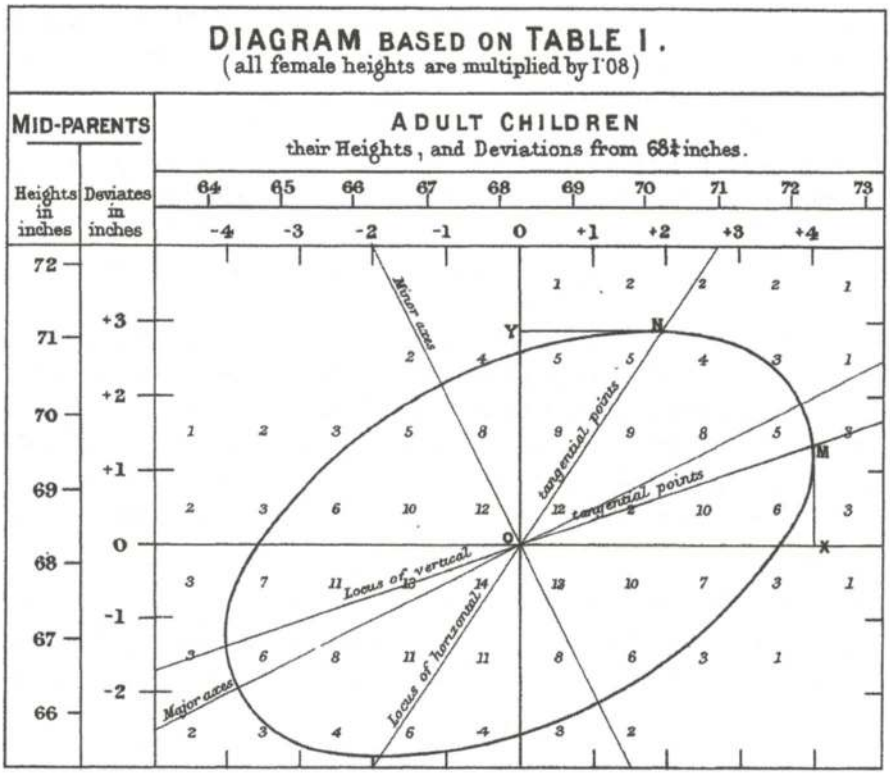


Figure 4.4. Galton's Regression Towards Mediocrity. Source: Francis Galton (1886) "Regression towards mediocrity in hereditary stature." *Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263.

perceived competence score and the two-party vote share differential is 0.43, whereas the standard deviations of X and Y are 0.19 and 0.29, respectively. Thus, an increase in the perceived competence score of 0.19 is associated with an average increase in the two-party vote share differential of 12–13 percentage points ($\approx 0.43 \times 0.29$).

The estimated **slope coefficient** from a linear regression model equals the ρ standard deviation unit increase in the outcome variable that is associated with an increase of 1 standard deviation in the predictor, where ρ is the correlation between the two variables.

4.2.4 REGRESSION TOWARDS THE MEAN

In his 1886 paper entitled "Regression towards mediocrity in hereditary stature," a British scholar, Sir Francis Galton, conducted one of the first regression analyses. He studied human hereditary stature by examining the relationship between the height of adult children and the average of their parents' heights, which Galton called the "mid-parents' height." Galton was the first to present an example of the phenomenon called *regression towards the mean*. He summarized this as "When Mid Parents are shorter (taller) than mediocrity, their Children tend to be taller (shorter) than they."

Figure 4.4 is taken from the original paper. In this figure, the values indicate the number of observations and the ellipse represents the data cloud. The "locus of

vertical tangential points" represents a regression line where the outcome variable is adult children's height (horizontal axis) and the predictor is their mid-parents' height (vertical axis). Note that the outcome variable is measured on the horizontal axis while the predictor is on the vertical axis, which is the exact opposite of the current practice of plotting the outcome variable on the vertical axis. Galton also regressed mid-parents' heights on the heights of adult children. This regression line is denoted by the "locus of horizontal tangential points." The angle of the slope of this regression line, which Galton calculated to be 2/3, represents the rate of regression from mid-parents to children.

To demonstrate the regression effect numerically, consider the observations that have mid-parents' heights of approximately 71 inches. As we can see from figure 4.4, there are 24 such observations, represented by those in the second row from the top. Out of these observations, only 8, or 33% of them, have children who are at least as tall as their mid-parents. In contrast, focus on the observations whose mid-parents are about 67 inches and hence shorter than the average height (they are in the second row from the bottom). Out of 57 such observations, 40 observations, or 70%, have children whose heights are at least their mid-parents' height. Galton called this pattern the "regression towards mediocrity." Note, however, that as indicated by the positive slope of the regression line, children whose parents are taller also tend to be taller on average. We emphasize that as shown in chapter 6 this empirical phenomenon can be explained by chance alone. Thus, regression towards the mean does not imply that human heights are converging and everyone will have an identical height in the future!

Regression towards the mean is observed in other contexts as well. Below, we show another example of this phenomenon, demonstrating that Obama tended to gain fewer votes in 2012 than in 2008 for the states in which he did well in 2008. Other examples include test scores where students who perform well in the midterm exam tend not to do as well in the final exam. An important point is that this decline in performance may have arisen due to chance rather than to a lack of Obama's or the students' efforts.

Regression towards the mean represents an empirical phenomenon where an observation with a value of the predictor further away from the distribution's mean tends to have a value of an outcome variable closer to that mean. This tendency can be explained by chance alone.

4.2.5 MERGING DATA SETS IN R

We will examine whether or not the US presidential election data exhibit the regression towards the mean phenomenon. To do this, we use Obama's vote share in the 2008 election to predict his vote share in his 2012 reelection. We *merge* the 2012 election result data set, `pres12.csv`, into the 2008 election data set. The variable names and descriptions of the 2012 election result data set are given in table 4.5.

Merging two data sets can be done in R using the `merge()` function. The function takes three main arguments, `x`, `y`, and `by`, where the `x` and `y` arguments represent two data frames to be merged and the `by` argument indicates the variable name(s) used for merging. Let's first look at two data sets we would like to merge.

Table 4.5. 2012 US Presidential Election Data.

Variable	*Description*
`state`	abbreviated name of the state
`Obama`	Obama's vote share (percentage)
`Romney`	Romney's vote share (percentage)
`EV`	number of Electoral College votes for the state

```
pres12 <- read.csv("pres12.csv")  # load 2012 data
## quick look at two data sets
head(pres08)
##   state.name state Obama McCain EV margin
## 1    Alabama    AL    39     60  9    -21
## 2     Alaska    AK    38     59  3    -21
## 3    Arizona    AZ    45     54 10     -9
## 4   Arkansas    AR    39     59  6    -20
## 5 California    CA    61     37 55     24
## 6   Colorado    CO    54     45  9      9
head(pres12)
##   state Obama Romney EV
## 1    AL    38     61  9
## 2    AK    41     55  3
## 3    AZ    45     54 11
## 4    AR    37     61  6
## 5    CA    60     37 55
## 6    CO    51     46  9
```

We will use the state name variable `state`, which is contained in both data sets, to merge the two data frames.

```
## merge two data frames
pres <- merge(pres08, pres12, by = "state")
## summarize the merged data frame
summary(pres)
##      state         state.name    Obama.x
##  AK     : 1   Alabama   : 1   Min.   :33.00
##  AL     : 1   Alaska    : 1   1st Qu.:43.00
##  AR     : 1   Arizona   : 1   Median :51.00
##  AZ     : 1   Arkansas  : 1   Mean   :51.37
##  CA     : 1   California: 1   3rd Qu.:57.50
```

```
##  CO      : 1   Colorado  : 1   Max.    :92.00
##  (Other):45   (Other)   :45
##       McCain           EV.x           margin
##  Min.    : 7.00   Min.   : 3.00   Min.   :-32.000
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##     Obama.y          Romney          EV.y
##  Min.   :25.00   Min.   : 7.00   Min.   : 3.00
##  1st Qu.:40.50   1st Qu.:41.00   1st Qu.: 4.50
##  Median :51.00   Median :48.00   Median : 8.00
##  Mean   :49.06   Mean   :49.04   Mean   :10.55
##  3rd Qu.:56.00   3rd Qu.:58.00   3rd Qu.:11.50
##  Max.   :91.00   Max.   :73.00   Max.   :55.00
##
```

Note that if the data frames have variables with identical names, i.e., `Obama` and `EV`, then the merged data frame will append `.x` and `.y` to each name, thereby attributing each variable to its original data frame. The variable used for merging must exist in both data frames. This variable may have the same name in both data frames, as in the above code chunk, but if the variable happens to have different names, then we can use the `by.x` and `by.y` arguments to specify the exact variable names used in each data frame. By default, the merged data frame keeps the name of the variable from data frame `x`, which is specified by the `by.x` argument. An example code chunk is given here.

```
## change the variable name for illustration
names(pres12)[1] <- "state.abb"
## merging data sets using the variables of different names
pres <- merge(pres08, pres12, by.x = "state", by.y = "state.abb")
summary(pres)
```

```
##      state          state.name    Obama.x
##  AK      : 1   Alabama   : 1   Min.   :33.00
##  AL      : 1   Alaska    : 1   1st Qu.:43.00
##  AR      : 1   Arizona   : 1   Median :51.00
##  AZ      : 1   Arkansas  : 1   Mean   :51.37
##  CA      : 1   California: 1   3rd Qu.:57.50
##  CO      : 1   Colorado  : 1   Max.   :92.00
##  (Other):45   (Other)   :45
##       McCain           EV.x           margin
##  Min.    : 7.00   Min.   : 3.00   Min.   :-32.000
```

```
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##     Obama.y          Romney          EV.y
##  Min.   :25.00   Min.   : 7.00   Min.   : 3.00
##  1st Qu.:40.50   1st Qu.:41.00   1st Qu.: 4.50
##  Median :51.00   Median :48.00   Median : 8.00
##  Mean   :49.06   Mean   :49.04   Mean   :10.55
##  3rd Qu.:56.00   3rd Qu.:58.00   3rd Qu.:11.50
##  Max.   :91.00   Max.   :73.00   Max.   :55.00
##
```

An alternative way of combining two data frames is the `cbind()` function, which enables column-binding of multiple data frames. (As a side note, the `rbind()` function performs row-binding of multiple data frames by stacking one below another.) But sometimes problematically, the `cbind()` function assumes the proper sorting of data frames such that corresponding observations appear in the same row of the data frames. In our current application, each state must appear in the same row of the two data frames. The `merge()` function, on the other hand, appropriately sorts the data frames according to the variable used for merging. Another disadvantage of the `cbind()` function is that it preserves all columns in both data frames even when they represent the same variable, containing identical information.

The code chunk below illustrates these two problems. The resulting merged data frame keeps all variables from both data frames, and more importantly, the merged data frame has incorrect information for the District of Columbia (`DC`) and Delaware (`DE`) because their order is different in the two original data frames. In contrast, the `merge()` function will sort the second data frame, `pres12`, appropriately to match with the first data frame, `pres08`.

```
## cbinding two data frames
pres1 <- cbind(pres08, pres12)
## this shows all variables are kept
summary(pres1)
##       state.name     state        Obama
##  Alabama    : 1   AK     : 1   Min.   :33.00
##  Alaska     : 1   AL     : 1   1st Qu.:43.00
##  Arizona    : 1   AR     : 1   Median :51.00
##  Arkansas   : 1   AZ     : 1   Mean   :51.37
##  California: 1   CA     : 1   3rd Qu.:57.50
##  Colorado   : 1   CO     : 1   Max.   :92.00
##  (Other)    :45   (Other):45
```

```
##      McCain            EV           margin
##  Min.   : 7.00   Min.   : 3.00   Min.   :-32.000
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##    state.abb      Obama           Romney
##  AK      : 1   Min.   :25.00   Min.   : 7.00
##  AL      : 1   1st Qu.:40.50   1st Qu.:41.00
##  AR      : 1   Median :51.00   Median :48.00
##  AZ      : 1   Mean   :49.06   Mean   :49.04
##  CA      : 1   3rd Qu.:56.00   3rd Qu.:58.00
##  CO      : 1   Max.   :91.00   Max.   :73.00
##  (Other):45
##        EV
##  Min.   : 3.00
##  1st Qu.: 4.50
##  Median : 8.00
##  Mean   :10.55
##  3rd Qu.:11.50
##  Max.   :55.00
##
## DC and DE are flipped in this alternative approach
pres1[8:9, ]
##   state.name state Obama McCain EV margin state.abb Obama
## 8       D.C.    DC    92      7  3     85        DE    59
## 9   Delaware    DE    62     37  3     25        DC    91
##   Romney EV
## 8     40  3
## 9      7  3
## merge() does not have this problem
pres[8:9, ]
##   state state.name Obama.x McCain EV.x margin Obama.y
## 8    DC       D.C.      92      7    3     85      91
## 9    DE   Delaware      62     37    3     25      59
##   Romney EV.y
## 8      7    3
## 9     40    3
```

Using the merged data frame, we investigate whether or not the regression towards the mean phenomenon exists in the US presidential election data. Given the recent

trend of increasing polarization in American politics (see section 3.5), we standardize vote shares across elections by computing their *z-scores* so that we can measure Obama's electoral performance in each state relative to his average performance of that year (see section 3.6.2). That is, we subtract the mean from Obama's vote share in each election and then divide it by the standard deviation. This can be done easily by using the `scale()` function. We perform this transformation because technically, the regression towards the mean phenomenon holds when both the outcome and explanatory variables are standardized.

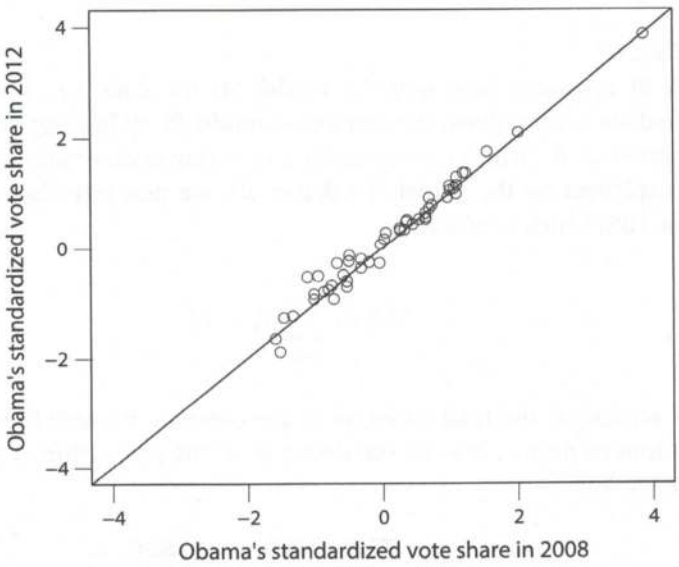
```
pres$Obama2008.z <- scale(pres$Obama.x)
pres$Obama2012.z <- scale(pres$Obama.y)
```

We regress Obama's 2012 standardized vote share on his 2008 standardized vote share. As expected, we observe a strong positive linear relationship between the two. Obama tended to receive more votes in 2012 from states that gave him more votes in 2008. Note that when we standardize both the outcome variable and the predictor, the estimated intercept becomes zero. This is because the estimated intercept is given by $\hat{\alpha} = \overline{Y} - \hat{\beta}\overline{X}$ (see equation (4.6)) and after standardizing, the sample means of both variables, $\overline{Y}$ and $\overline{X}$, are zero. As shown below, in this case, R estimates the intercept to be essentially zero. It is also possible to fit the model without an intercept by including `-1` in the formula.

```
## intercept is estimated as essentially zero
fit1 <- lm(Obama2012.z ~ Obama2008.z, data = pres)
fit1

##
## Call:
## lm(formula = Obama2012.z ~ Obama2008.z, data = pres)
##
## Coefficients:
## (Intercept)  Obama2008.z
##  -3.521e-17    9.834e-01

## regression without an intercept; estimated slope is identical
fit1 <- lm(Obama2012.z ~ -1 + Obama2008.z, data = pres)
fit1

##
## Call:
## lm(formula = Obama2012.z ~ -1 + Obama2008.z, data = pres)
##
## Coefficients:
## Obama2008.z
##      0.9834
```

Here, we plot the fitted regression line as well as the data points where we observe a strong linear relationship.

```
plot(pres$Obama2008.z, pres$Obama2012.z, xlim = c(-4, 4), ylim = c(-4, 4),
     xlab = "Obama's standardized vote share in 2008",
     ylab = "Obama's standardized vote share in 2012")
abline(fit1) # draw a regression line
```



Now we compute the proportion of states where Obama received a greater share of standardized votes in 2012 than he did in 2008. We do so using first the bottom quartile of Obama's 2008 (standardized) vote share, then the top quartile. If the regression towards the mean phenomenon exists, then this proportion should be greater for the states in the bottom quartile than those in the top quartile.

```
## bottom quartile
mean((pres$Obama2012.z >
        pres$Obama2008.z)[pres$Obama2008.z
                          <= quantile(pres$Obama2008.z, 0.25)])
## [1] 0.5714286
## top quartile
mean((pres$Obama2012.z >
        pres$Obama2008.z)[pres$Obama2008.z
                          >= quantile(pres$Obama2008.z, 0.75)])
## [1] 0.4615385
```

In the above code, we use the `quantile()` function to compute the top and bottom quartiles. Then, a logical vector where `TRUE` (`FALSE`) indicates Obama's 2012 vote share being greater than (less than or equal to) his 2008 vote share is subsetted by another logical vector. This second logical vector, inside the square brackets, indicates whether Obama's 2008 vote share for a state is in the bottom or top quartile. The result clearly shows the regression towards the mean phenomenon. Obama fared better in 2012 than in 2008 in 57% of bottom quartile states, where he failed most in 2008. In contrast, Obama fared better in 2012 only among 46% of the top quartile states, where he succeeded most in 2008.

4.2.6 MODEL FIT

Model fit measures how well the model fits the data, i.e., how accurately the model predicts observations. We can assess model fit by looking at the *coefficient of determination*, or R^2, which represents the proportion of total variation in the outcome variable explained by the model. To define R^2, we first introduce the *total sum of squares* or TSS, which is defined as

$$\text{TSS} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2.$$

The TSS represents the total variation of the outcome variable based on the square distance from its mean. Now, we can define R^2 as the proportion of TSS explained by the predictor X:

$$R^2 = \frac{\text{TSS} - \text{SSR}}{\text{TSS}} = 1 - \frac{\text{SSR}}{\text{TSS}}.$$

The SSR or sum of squared residuals is defined in equation (4.4) and represents the residual variation of Y left unexplained by X. The value of R^2 ranges from 0 (when the correlation between the outcome and the predictor is 0) to 1 (when the correlation is 1), indicating how well the linear model fits the data at hand.

> The **coefficient of determination** is a measure of model fit and represents the proportion of variation in the outcome variable explained by the predictor. It is defined as one minus the ratio of the sum of squared residuals (SSR) to the total sum of squares (TSS).

As an illustrative example, consider the problem of predicting the 2000 US election results in Florida using the 1996 US election results from the same state at the county level. In Florida, there are 68 counties, and the CSV file `florida.csv` contains the number of votes cast for each candidate in those two elections. Table 4.6 displays the names and descriptions of variables in this data file. We focus on libertarian candidates

Table 4.6. 1996 and 2000 US Presidential Election Data for Florida Counties.

Variable	*Description*
`county`	county name
`Clinton96`	Clinton's votes in 1996
`Dole96`	Dole's votes in 1996
`Perot96`	Perot's votes in 1996
`Bush00`	Bush's votes in 2000
`Gore00`	Gore's votes in 2000
`Buchanan00`	Buchanan's votes in 2000

Ross Perot in 1996 and Pat Buchanan in 2000, using the votes for the former to predict the votes for the latter. We then compute R^2 from this regression model by first computing TSS and then SSR. Recall that the `resid()` function extracts the vector of residuals from the regression output.

```
florida <- read.csv("florida.csv")
## regress Buchanan's 2000 votes on Perot's 1996 votes
fit2 <- lm(Buchanan00 ~ Perot96, data = florida)
fit2

##
## Call:
## lm(formula = Buchanan00 ~ Perot96, data = florida)
##
## Coefficients:
## (Intercept)      Perot96
##     1.34575      0.03592

## compute TSS (total sum of squares) and SSR (sum of squared residuals)
TSS2 <- sum((florida$Buchanan00 - mean(florida$Buchanan00))^2)
SSR2 <- sum(resid(fit2)^2)
## coefficient of determination
(TSS2 - SSR2) / TSS2

## [1] 0.5130333
```

The result shows that 51% of the variation of Buchanan's 2000 votes can be explained by Perot's 1996 votes.

We turn this calculation into a function so that we can easily compute the coefficient of determination for different regression models (see section 1.3.4). The function takes as input the output from the `lm()` function, which is a list object containing various elements (see section 3.7.2). The value of the outcome variable can be recomputed from

the regression output object by summing the fitted value, which can be obtained using the `fitted()` function, and the residual for each observation.

```
R2 <- function(fit) {
    resid <- resid(fit) # residuals
    y <- fitted(fit) + resid # outcome variable
    TSS <- sum((y - mean(y))^2)
    SSR <- sum(resid^2)
    R2 <- (TSS - SSR) / TSS
    return(R2)
}
R2(fit2)
## [1] 0.5130333
```

Alternatively, we can obtain the value of R^2 by applying the `summary()` function to the output from the `lm()` function (see also section 7.3).

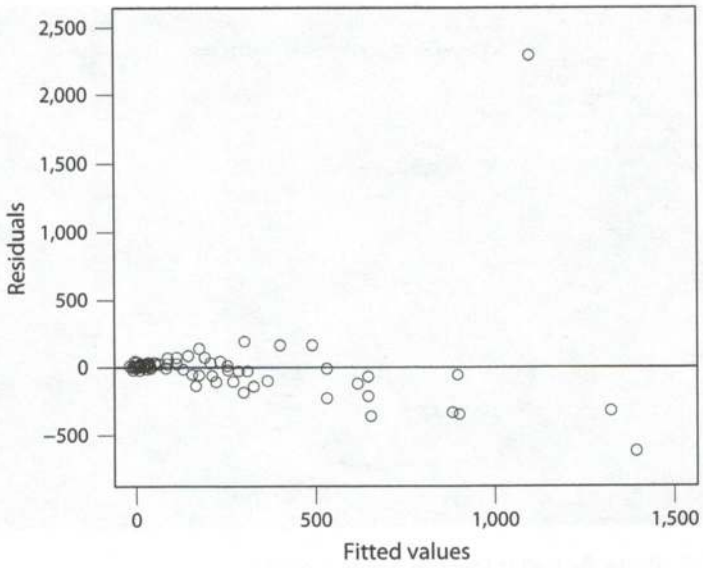
```
## built-in R function
fit2summary <- summary(fit2)
fit2summary$r.squared
## [1] 0.5130333
```

The resulting coefficient of determination appears relatively low given that we are predicting votes for a candidate from the same party using the previous election result. Earlier, we saw that Obama's vote shares at the state level are strongly correlated between the 2008 and 2012 elections. We can compute R^2 for that regression where the corresponding output object is `fit1`, which represents the output of the state-level regression. The coefficient of determination for the Florida regression proves to be much lower than that for the state-level regression.

```
R2(fit1)
## [1] 0.9671579
```

Given this unusually poor performance, it is useful to more closely inspect the residuals from the Florida regression. To do this, we create a *residual plot* where residuals are plotted against fitted values.

```
plot(fitted(fit2), resid(fit2), xlim = c(0, 1500), ylim = c(-750, 2500),
     xlab = "Fitted values", ylab = "Residuals")
abline(h = 0)
```



We observe an extremely large residual or *outlier*, where in the 2000 election, Buchanan received 2000 votes, substantially more than expected. The next line of code shows that this observation represents Palm Beach county. This can be seen by extracting the county name whose residual equals the maximum value of residuals.

```
florida$county[resid(fit2) == max(resid(fit2))]
## [1] PalmBeach
## 67 Levels: Alachua Baker Bay Bradford Brevard ... Washington
```

It turns out that in Palm Beach county, the so-called *butterfly ballot* was used for this election. A picture of this ballot is shown in figure 4.5. Voters are supposed to punch a hole that corresponds to the candidate they would like to vote for. However, as can be seen in the picture, the ballot is quite confusing. It appears that many supporters of Al Gore in this county mistakenly voted for Buchanan by punching the second hole from the top instead of the third hole. As mentioned at the beginning of the chapter, in the 2000 election, George Bush was elected to office by winning Florida with a razor-thin margin of 537 votes even though Gore won over half a million votes more than Bush in the entire country. It is widely believed that voting irregularities in Palm Beach county, as evident in the residual plot, cost Gore the presidency.

We now fit the same model without Palm Beach county. Later, we will see whether this removal improves the model fit, by comparing residual plots and regression lines with Palm Beach against those without it. We begin by computing the coefficient of determination without Palm Beach.

```
## data without Palm Beach
florida.pb <- subset(florida, subset = (county != "PalmBeach"))
```

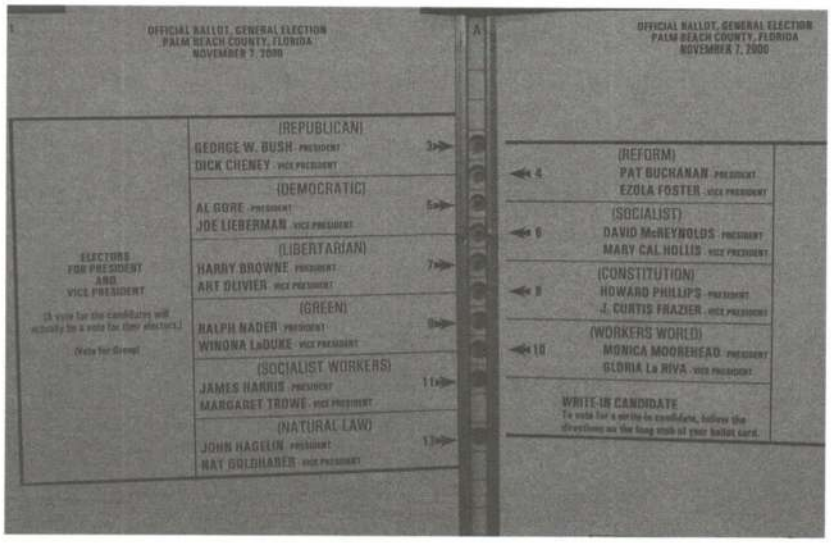


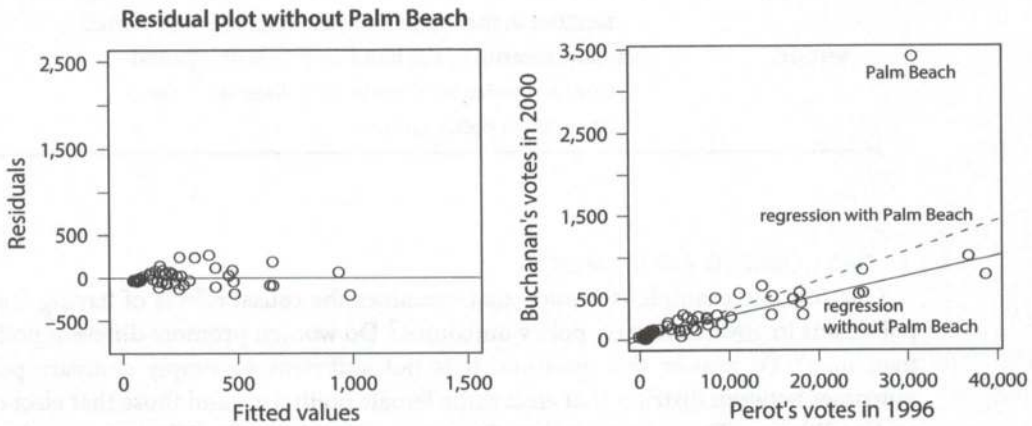
Figure 4.5. Butterfly Ballot in Palm Beach County.

```
fit3 <- lm(Buchanan00 ~ Perot96, data = florida.pb)
fit3

##
## Call:
## lm(formula = Buchanan00 ~ Perot96, data = florida.pb)
##
## Coefficients:
## (Intercept)      Perot96
##    45.84193      0.02435

## R-squared or coefficient of determination
R2(fit3)

## [1] 0.8511675
```

Without Palm Beach, the coefficient of determination dramatically increases from 0.51 to 0.85. The improvement in model fit can also be easily seen through the residual plot as well as the scatter plot with regression lines. We find that the regression line is influenced by Palm Beach—removing it shifts the regression line considerably. The new regression line fits the remaining observations better.

```
## residual plot
plot(fitted(fit3), resid(fit3), xlim = c(0, 1500), ylim = c(-750, 2500),
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residual plot without Palm Beach")
abline(h = 0) # horizontal line at 0
```

```
plot(florida$Perot96, florida$Buchanan00, xlab = "Perot's votes in 1996",
     ylab = "Buchanan's votes in 2000")
abline(fit2, lty = "dashed") # regression with Palm Beach
abline(fit3) # regression without Palm Beach
text(30000, 3250, "Palm Beach")
text(30000, 1500, "regression\n with Palm Beach")
text(30000, 400, "regression\n without Palm Beach")
```



Finally, it is important to emphasize that the model fit considered in this section is based on *in-sample predictions* rather than *out-of-sample predictions*. That is, model fit statistics, such as the coefficient of determination, describe how well one's model fits the sample at hand. If tailored too closely to a particular sample, which is called *overfitting*, the model may make less accurate predictions in another sample. In cases where we seek a general model that can be applied to other data, we need to be careful to avoid overfitting the model to a particular sample. In section 4.3.2, we will describe one way to adjust R^2 in order to reduce the possibility of overfitting.

4.3 Regression and Causation

Regression is a primary tool for making predictions in social science research. How can regression be used to draw causal inference? As we discussed in chapter 2, causal inference requires the prediction of counterfactual outcomes. For example, for units who received a treatment, we wish to predict the values of the outcome variable that would result without the treatment. Under certain assumptions, regression models can be used to predict counterfactual outcomes. We must be careful, however, because association, which can be quantified through regression, does not necessarily imply causation.

Table 4.7. Women as Policy Makers Data.

Variable	*Description*
`GP`	identifier for the Gram Panchayat (GP)
`village`	identifier for each village
`reserved`	binary variable indicating whether the GP was reserved for women leaders or not
`female`	binary variable indicating whether the GP had a female leader or not
`irrigation`	variable measuring the number of new or repaired irrigation facilities in the village since the reserve policy started
`water`	variable measuring the number of new or repaired drinking water facilities in the village since the reservation policy started

4.3.1 RANDOMIZED EXPERIMENTS

Our running example is a study that examines the causal effects of having female politicians in government on policy outcomes.[5] Do women promote different policies than men? To answer this question, it is not sufficient to simply compare policy outcomes between districts that elect some female politicians and those that elect only male politicians. This is because these two types of districts may differ in terms of many factors other than having female politicians. For example, if liberal districts may be more likely to elect female politicians, it is not clear whether policy differences can be attributed to ideology or politician's gender.

To overcome this potential confounding problem, the authors of the study took advantage of a randomized policy experiment in India where, since the mid-1990s, one-third of village council heads have been randomly reserved for female politicians. The CSV data set `women.csv` contains a subset of this data from West Bengal. The policy was implemented at the level of government called Gram Panchayat or GP. Each GP contains many villages. For this study, two villages were selected at random within each GP for detailed data collection. Table 4.7 shows the names and descriptions of the variables in this data set. Each observation in the data set represents a village and there are two villages associated with each GP.

We first check whether or not the reservation policy was properly implemented by computing the proportions of female politicians elected for the reserved seats as well as the unreserved ones. Since each GP has the same number of villages, we can simply compute the average across villages without creating a new data set at the GP level. For the reserved seats, this proportion should be equal to 1.

[5] This section is based on Raghabendra Chattopadhyay and Esther Duflo (2004) "Women as policy makers: Evidence from a randomized policy experiment in India." *Econometrica*, vol. 72, no. 5, pp. 1409–1443.

```
women <- read.csv("women.csv")
## proportion of female politicians in reserved GP vs. unreserved GP
mean(women$female[women$reserved == 1])
## [1] 1
mean(women$female[women$reserved == 0])
## [1] 0.07476636
```

It appears that the reservation policy has been followed. Every GP that was supposed to reserve a council position for women actually elected at least one female politician. In contrast, 93% of the GPs to which the reservation policy was not applicable had no female representative. Following what we learned in chapter 2, we can compare the mean policy outcomes between the villages in the reserved GPs and those in the unreserved GPs. We hypothesize that female politicians are more likely to support policies that female voters want. The researchers found that more women complain about the quality of drinking water than men, who more frequently complain about irrigation. We estimate the average causal effects of the reservation policy on the number of new or repaired irrigation systems and drinking water facilities in the villages since the policy was implemented. We use the difference-in-means estimator as in section 2.4.

```
## drinking water facilities
mean(women$water[women$reserved == 1]) -
    mean(women$water[women$reserved == 0])
## [1] 9.252423
## irrigation facilities
mean(women$irrigation[women$reserved == 1]) -
    mean(women$irrigation[women$reserved == 0])
## [1] -0.3693319
```

We find that the reservation policy increased the number of drinking water facilities in a GP on average by about 9 (new or repaired), whereas the policy had little effect on irrigation systems. This finding is consistent with the aforementioned hypothesis that female politicians tend to represent the interests of female voters.

How can we use regression to analyze the data from randomized experiments like this one? It turns out that regressing an outcome variable on a treatment variable yields a slope coefficient identical to the difference in average outcomes between the two groups. In addition, the resulting intercept corresponds to the average outcome among the control units. More generally, when the predictor X is binary, taking a value of either 0 or 1, the linear model defined in equation (4.1) yields the estimated coefficients

of the following expressions:

$$\hat{\alpha} = \underbrace{\frac{1}{n_0}\sum_{i=1}^{n}(1 - X_i)Y_i}_{\text{mean outcome among the control}},$$

$$\hat{\beta} = \underbrace{\frac{1}{n_1}\sum_{i=1}^{n} X_i Y_i}_{\text{mean outcome among the treated}} - \underbrace{\frac{1}{n_0}\sum_{i=1}^{n}(1 - X_i)Y_i}_{\text{mean outcome among the control}}$$

In this equation, $n_1 = \sum_{i=1}^{n} X_i$ is the size of the treatment group and $n_0 = n - n_1$ is the size of the control group. Thus, $\hat{\beta}$ can be interpreted as the estimated average treatment effect.

Using our experimental data, we confirm this numerical equivalence between regression coefficients and average outcomes. That is, we observe that the estimated slope coefficient is equal to the corresponding *difference-in-means estimator*.

```
lm(water ~ reserved, data = women)

##
## Call:
## lm(formula = water ~ reserved, data = women)
##
## Coefficients:
## (Intercept)     reserved
##      14.738        9.252

lm(irrigation ~ reserved, data = women)

##
## Call:
## lm(formula = irrigation ~ reserved, data = women)
##
## Coefficients:
## (Intercept)     reserved
##      3.3879      -0.3693
```

We can directly connect the potential outcomes covered in chapter 2 to the regression model:

$$Y(X) = \alpha + \beta X + \epsilon.$$

Since the regression model predicts the average outcome given a value of the predictor, the estimated average treatment effect equals the estimated slope coefficient when X is binary. Recall that $\hat{\beta}$ represents the estimated change in Y when X is increased by one unit. Then, we have $\widehat{Y(1)} - \widehat{Y(0)} = (\hat{\alpha} + \hat{\beta}) - \hat{\alpha} = \hat{\beta}$, while the estimated average outcome for the control group is equal to the estimated intercept, i.e., $\widehat{Y(0)} = \hat{\alpha}$. Thus,

the linear regression model provides an alternative, but numerically equivalent, way to analyze experimental data in this setting.

When applied to experimental data with a single, binary treatment, the estimated slope coefficient of the linear regression model can be interpreted as an estimate of average treatment effect and is numerically equivalent to the **difference-in-means estimator**. The estimated intercept, on the other hand, is equal to the estimated average outcome under the control condition. The randomization of treatment assignment permits this **causal interpretation** of association identified under a linear regression model.

4.3.2 REGRESSION WITH MULTIPLE PREDICTORS

So far, we have included only one predictor in the linear regression model. However, a regression model can have more than one predictor. In general, a linear regression model with multiple predictors is defined as

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon.$$

In this model, α is the intercept, β_j is the coefficient for predictor X_j, ϵ is an error term, and p is the number of predictors and can be greater than 1. The interpretation of each coefficient β_j is the amount of change in the outcome variable associated with a one-unit increase in the corresponding predictor X_j *when all other predictors are held constant* or so-called *ceteris paribus*. Therefore, linear regression with multiple predictors enables researchers to assess the impact of each predictor.

The least squares method, as described in section 4.2.3, can be used to estimate the model parameters. That is, we choose the values of $(\hat{\alpha}, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ such that the *sum of squared residuals* (SSR) is minimized. The SSR is defined as

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta}_1 X_{i1} - \hat{\beta}_2 X_{i2} - \cdots - \hat{\beta}_p X_{ip})^2.$$

In the equation, $\hat{\epsilon}_i$ is the *residual* and X_{ij} is the value of the jth predictor for the ith observation. Recall that the residual is defined as the difference between the observed response Y and its predicted or fitted value $\widehat{Y} = \hat{\alpha} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta} X_p$.

The validity of predictions based on a linear regression model critically rests on the assumption of linearity. The method of least squares always gives us the line that "best fits" the data in the sense of minimizing the SSR. However, this does not necessarily mean that the linear model is appropriate. While a comprehensive treatment of testing and relaxing this assumption is beyond the scope of this book, we must not forget that any model or method requires an assumption, and linear regression is no exception.

As an example of linear regression models with multiple predictors, we consider the randomized experiment on social pressure and turnout introduced in section 2.4.2. In that study, registered voters were randomly assigned to one of the four groups. We can fit a linear regression model, in which group assignment is used to predict turnout. Fitting the linear regression model is done via the `lm()` function as before.

One can add more than one predictor by simply using the + operator, for example, `lm(y ~ x1 + x2 + x3)`. In this example, since the `messages` variable is a factor, the `lm()` function automatically creates a set of *indicator* or dummy variables, each of which is equal to 1 if a voter is assigned to the corresponding group. These indicator variables will be used for computation but will not be saved in the data frame. The model includes all but the variable corresponding to the base level. The base level of a factor variable is the level displayed first when we apply the `levels()` function, which lists levels in alphabetical order. The other values of a factor variable are defined in relation to this base level value.

```
social <- read.csv("social.csv")
levels(social$messages) # base level is "Civic Duty"
## [1] "Civic Duty" "Control"    "Hawthorne"    "Neighbors"
```

Now we fit the linear regression model using this factor variable.

```
fit <- lm(primary2008 ~ messages, data = social)
fit
##
## Call:
## lm(formula = primary2008 ~ messages, data = social)
##
## Coefficients:
##       (Intercept)    messagesControl    messagesHawthorne
##          0.314538          -0.017899           0.007837
## messagesNeighbors
##          0.063411
```

Alternatively, one can create an indicator variable for each group and then specify the regression model using them. The results are identical to those given above.

```
## create indicator variables
social$Control <- ifelse(social$messages == "Control", 1, 0)
social$Hawthorne <- ifelse(social$messages == "Hawthorne", 1, 0)
social$Neighbors <- ifelse(social$messages == "Neighbors", 1, 0)
## fit the same regression as above by directly using indicator variables
lm(primary2008 ~ Control + Hawthorne + Neighbors, data = social)
```

Mathematically, the linear regression model we just fit is given by

$$Y = \alpha + \beta_1 \,\mathsf{Control} + \beta_2 \,\mathsf{Hawthorne} + \beta_3 \,\mathsf{Neighbors} + \epsilon.$$

In this model, each predictor is an indicator variable for the corresponding group. Since the base level of the `messages` variable is `"Civic Duty"`, the `lm()` function

excludes the corresponding indicator variable. Using the fitted model, we can predict the average outcome, which in this case is the average proportion of voters who turned out. For example, under the `Control` condition, the average outcome is predicted to be $\hat{\alpha} + \hat{\beta}_1 = 0.315 + (-0.018) = 0.297$ or 29.7%. Similarly, for the `Neighbors` group, the predicted average outcome is $\hat{\alpha} + \hat{\beta}_3 = 0.315 + 0.063 = 0.378$.

The predicted average outcome can be obtained using the `predict()` function. This function, like the `fitted()` function, takes the output from the `lm()` function and computes predicted values. However, unlike the `fitted()` function, which computes predicted values for the sample used to fit the model, the `predict()` function can take a new data frame as the `newdata` argument and make predictions for each observation in this data frame. The new data frame's variables must match the predictors of the fitted linear model, though they can have different values. In the current application, we create a new data frame using the `data.frame()` function. The resulting data frame contains the same variable `messages` as the predictor of the model but only four observations, each of which has one of the unique values of the original `messages` variable. We use the `unique()` function to extract these unique values and return them in the order of their first occurrence.

```
## create a data frame with unique values of "messages"
unique.messages <- data.frame(messages = unique(social$messages))
unique.messages

##     messages
## 1 Civic Duty
## 2  Hawthorne
## 3    Control
## 4  Neighbors

## make prediction for each observation from this new data frame
predict(fit, newdata = unique.messages)

##         1         2         3         4
## 0.3145377 0.3223746 0.2966383 0.3779482
```

As we saw in the case of a linear regression model with a single, binary predictor (see section 4.3.1), the predicted average outcome for each treatment condition equals the sample average within the corresponding subset of the data.

```
## sample average
tapply(social$primary2008, social$messages, mean)

## Civic Duty    Control  Hawthorne  Neighbors
##  0.3145377  0.2966383  0.3223746  0.3779482
```

To make the output of linear regression more interpretable, we can remove an intercept and use all four indicator variables (rather than removing the indicator variable for the base level in order to include a common intercept). This alternative

specification enables us to directly obtain the average outcome within each group as a coefficient for the corresponding indicator variable. To omit the intercept in linear regression, we simply use `-1` in the formula. The following code chunk illustrates this.

```
## linear regression without intercept
fit.noint <- lm(primary2008 ~ -1 + messages, data = social)
fit.noint

##
## Call:
## lm(formula = primary2008 ~ -1 + messages, data = social)
##
## Coefficients:
## messagesCivic Duty     messagesControl   messagesHawthorne
##             0.3145              0.2966              0.3224
##  messagesNeighbors
##             0.3779
```

Each coefficient above represents the average outcome for a given group. As a result, we can estimate an average treatment effect relative to the control for each treatment condition (`Civic Duty`, `Hawthorne`, or `Neighbors`) by calculating that treatment condition's coefficient minus the coefficient for the control group, which is the baseline group under this model with no intercept. The difference in the estimated causal effects between any two groups equals the difference between the corresponding coefficients, whether one uses the model with no intercept or the original model. Therefore, the average effect of the `Neighbors` treatment (relative to the `Control` condition) equals $0.378 - 0.297$ in the model with no intercept, or $0.063 - (-0.018)$ in the original model, either of which equals 0.081 or 8.1 percentage points. As was the case before, the same estimate of average causal effect can be obtained in two ways—through linear regression with a factor treatment variable or the difference-in-means estimator.

```
## estimated average effect of "Neighbors" condition
coef(fit)["messagesNeighbors"] - coef(fit)["messagesControl"]

## messagesNeighbors
##        0.08130991

## difference-in-means
mean(social$primary2008[social$messages == "Neighbors"]) -
    mean(social$primary2008[social$messages == "Control"])

## [1] 0.08130991
```

Finally, we can compute the *coefficient of determination* or R^2 as in section 4.2.6. When there are multiple predictors, however, we often compute the *adjusted* R^2 with the so-called *degrees of freedom* correction that accounts for the number of predictors.

Roughly speaking, the degrees of freedom refers to the number of observations that are "free to vary," which is often represented by the total number of observations minus the number of parameters to be estimated. In the current setting, the degrees of freedom equals $n - p - 1 = n - (p + 1)$ because n is the number of observations and $p + 1$ is the number of coefficients to be estimated, i.e., a coefficient for each of p predictors plus an intercept.

Since one can always increase the (unadjusted) R^2 by including an additional predictor (which always decreases SSR), the degrees of freedom correction adjusts R^2 downwards as more predictors are included in the model. The formula of the adjusted R^2 is given by

$$\text{adjusted } R^2 = 1 - \frac{\text{SSR}/(n - p - 1)}{\text{TSS}/(n - 1)}.$$

SSR is divided by the number of observations n minus the number of coefficients to be estimated $(p + 1)$. TSS is divided by $(n - 1)$ since TSS estimates only one parameter, the mean of the outcome variable or $\overline{Y}$. As in section 4.2.6, we create a function that computes the adjusted R^2.

```
## adjusted R-squared
adjR2 <- function(fit) {
    resid <- resid(fit) # residuals
    y <- fitted(fit) + resid # outcome
    n <- length(y)
    TSS.adj <- sum((y - mean(y))^2) / (n - 1)
    SSR.adj <- sum(resid^2) / (n - length(coef(fit)))
    R2.adj <- 1 - SSR.adj / TSS.adj
    return(R2.adj)
}
adjR2(fit)

## [1] 0.003272788

R2(fit) # unadjusted R-squared calculation

## [1] 0.003282564
```

In this case, the difference between unadjusted and adjusted R^2 is small because the number of observations is large relative to the number of coefficients. Alternatively, we can obtain both adjusted and unadjusted R^2 by applying the `summary()` function to output from the `lm()` function (see also section 7.3).

```
fitsummary <- summary(fit)
fitsummary$adj.r.squared

## [1] 0.003272788
```

The **linear regression model with multiple predictors** is defined as

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

where the coefficient β_j represents the increase in the average outcome associated with a one-unit increase in X_j while holding the other variables constant. The coefficients are estimated by minimizing the sum of squared residuals. The **degrees of freedom** adjustment is often made when computing the coefficient of determination.

4.3.3 HETEROGENEOUS TREATMENT EFFECTS

When applied to randomized experiments, linear regression with multiple predictors can also be helpful for exploring *heterogeneous treatment effects*. Even if the average treatment effect is positive, for example, the same treatment may affect some individuals in a negative way. Identifying the characteristics associated with the direction and magnitude of the treatment effect is essential in determining who should receive the treatment. In the current application, we might hypothesize that the social pressure treatment would barely affect those who vote infrequently. In contrast, they may be the ones who would be most affected by such treatment. To illustrate the analysis of heterogeneous treatment effects, we examine the difference in the estimated average causal effect of the `Neighbors` message between those who voted in the 2004 primary election and those who did not. We can do this by subsetting the data and then estimating the average treatment effect within each subset. Finally, we compare these two estimated average treatment effects.

```
## average treatment effect (ATE) among those who voted in 2004 primary
social.voter <- subset(social, primary2004 == 1)
ate.voter <-
    mean(social.voter$primary2006[social.voter$messages == "Neighbors"]) -
        mean(social.voter$primary2006[social.voter$messages == "Control"])
ate.voter

## [1] 0.09652525

## average effect among those who did not vote
social.nonvoter <- subset(social, primary2004 == 0)
ate.nonvoter <-
    mean(social.nonvoter$primary2006[social.nonvoter$messages == "Neighbors"]) -
        mean(social.nonvoter$primary2006[social.nonvoter$messages == "Control"])
ate.nonvoter

## [1] 0.06929617
```

```
## difference
ate.voter - ate.nonvoter

## [1] 0.02722908
```

We find that those who voted in the 2004 primary election have the estimated average effect of 9.7 percentage points, which is approximately 2.7 percentage points greater than those who did not vote in the election. This implies that the `Neighbors` message affects those who voted in the 2004 primary election more than those who did not.

The same analysis can be carried out through the use of linear regression with an *interaction effect* between the treatment variable `Neighbors` and the covariate of interest `primary2004`. In our application, the model is given by

$$Y = \alpha + \beta_1 \text{primary2004} + \beta_2 \text{Neighbors} + \beta_3 (\text{primary2004} \times \text{Neighbors}) + \epsilon. \tag{4.9}$$

The final predictor is the product of two indicator variables, primary2004 × Neighbors, which is equal to 1 if and only if an individual voted in the 2004 primary election (`primary2004 = 1`) and received the `Neighbors` treatment (`Neighbors = 1`).

Thus, according to the model, among the voters who turned out in the 2004 primary election (`primary2004 = 1`), the average effect of the `Neighbors` message equals $\beta_2 + \beta_3$, whereas the same effect for those who did not vote in the 2004 election (`primary2004 = 0`) equals β_2. Thus, the coefficient for the interaction term β_3 represents the additional average treatment effect the first group of voters receive relative to the second group.

More generally, an example of the linear regression model with an interaction term is

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon,$$

where the coefficient for the interaction term β_3 represents how the effect of X_1 depends on X_2 (or vice versa). To see this, set $X_2 = x_2$ and then compute the predicted value when $X_1 = x_1$. This is given by $\hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_1 x_2$. Now, compare this with the predicted value when X_1 is increased by one unit, i.e., $X_1 = x_1 + 1$. Under this scenario, the predicted value is $\hat{\alpha} + \hat{\beta}_1 (x_1 + 1) + \hat{\beta}_2 x_2 + \hat{\beta}_3 (x_1 + 1) x_2$. Then, subtracting the previous predicted value from this one, we obtain the following expression for how the change in the average outcome associated with a one-unit increase in X_1 depends on the value of X_2:

$$\hat{\beta}_1 + \hat{\beta}_3 x_2.$$

This is another linear equation. The intercept β_1 represents the increase in the average outcome associated with a one-unit increase in X_1 when $X_2 = 0$. Then, each one-unit increase in X_2 has the effect of further increasing X_1 by the slope $\hat{\beta}_3$.

An example of a linear regression model with an **interaction term** is

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

The model assumes that the effect of X_1 linearly depends on X_2. That is, as we increase X_2 by one unit, the change in the average outcome associated with a one-unit increase of X_1 goes up by β_3.

In R, an interaction term can be represented by a colon `:` with the syntax `x1:x2` producing an interaction term between the two variables `x1` and `x2`. We illustrate the use of interaction terms by focusing on the `Neighbors` and `Control` groups.

```
## subset Neighbors and Control groups
social.neighbor <- subset(social, (messages == "Control") |
                                  (messages == "Neighbors"))
## standard way to generate main and interaction effects
fit.int <- lm(primary2006 ~ primary2004 + messages + primary2004:messages,
              data = social.neighbor)
fit.int

##
## Call:
## lm(formula = primary2006 ~ primary2004 + messages + primary2004:messages,
##     data = social.neighbor)
##
## Coefficients:
##                   (Intercept)
##                       0.23711
##                    primary2004
##                       0.14870
##              messagesNeighbors
##                       0.06930
## primary2004:messagesNeighbors
##                       0.02723
```

Since the `Control` group is the baseline condition, the slope coefficients are estimated only for the `Neighbors` condition and its interaction with the `primary2004` variable.

Alternatively, an asterisk `*` generates two *main effect* terms as well as one interaction effect term. That is, the syntax `x1*x2` produces `x1`, `x2`, and `x1:x2`. In most applications, one should include the corresponding main effects when the model has an interaction term. The same regression model as above can be fitted using the following syntax.

```
lm(primary2006 ~ primary2004 * messages, data = social.neighbor)
```

To interpret each estimated coefficient, it is again helpful to consider the predicted average outcome. Among those who voted in the 2004 primary election, the estimated average effect of the `Neighbors` treatment can be written as the difference in the estimated average outcome between the treatment and control groups. In terms of model parameters, this difference is equal to $(\hat{\alpha} + \hat{\beta}_1 + \hat{\beta}_2 + \hat{\beta}_3) - (\hat{\alpha} + \hat{\beta}_1) = \hat{\beta}_2 + \hat{\beta}_3$, where $\hat{\beta}_2$ and $\hat{\beta}_3$ are excluded from the second part of the equation because for the control group, `Neighbors` equals `0`. In contrast, the estimated average treatment effect among those who did not vote is given by $(\hat{\alpha} + \hat{\beta}_2) - \hat{\alpha} = \hat{\beta}_2$. Thus, the difference in the estimated average treatment effect between those who voted in the 2004 primary election and those who did not equals the estimated coefficient for the interaction effect term, i.e., $(\hat{\beta}_2 + \hat{\beta}_3) - \hat{\beta}_2 = \hat{\beta}_3$. This implies that the coefficient for the interaction effect term β_3 characterizes how the average treatment effect varies as a function of the covariate.

While we have so far focused on a factor or categorical variable, it is also possible to use a continuous variable as a predictor. The use of continuous variables requires a stronger linearity assumption that a one-unit increase in the predictor leads to an increase of the same size in the outcome, regardless of the baseline value. In the current application, we consider the age of the voter in 2006 as a predictor. We first compute this variable by subtracting the year of birth variable from the year of election.

```
social.neighbor$age <- 2006 - social.neighbor$yearofbirth
summary(social.neighbor$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   41.00   50.00   49.82   59.00  106.00
```

Thus, in this subset of the data, the ages of voters vary from 20 to 106. We now explore how the average causal effect of the `Neighbors` treatment changes as a function of age. To do this, we use the `age` variable instead of the `primary2004` variable in the linear regression model given in equation (4.9):

$$Y = \alpha + \beta_1 \,\text{age} + \beta_2 \,\text{Neighbors} + \beta_3 \,(\text{age} \times \text{Neighbors}) + \epsilon.$$

We can use the same computation strategy as above to understand how the average treatment effect changes as a function of age. Consider a group of voters who are x years old. The estimated average treatment effect of the `Neighbors` message for these voters is given by $(\hat{\alpha} + \hat{\beta}_1 x + \hat{\beta}_2 + \hat{\beta}_3 x) - (\hat{\alpha} + \hat{\beta}_1 x) = \hat{\beta}_2 + \hat{\beta}_3 x$. In contrast, among the voters who are $(x + 1)$ years old, the estimated average effect is $\{\hat{\alpha} + \hat{\beta}_1(x + 1) + \hat{\beta}_2 + \hat{\beta}_3(x + 1)\} - \{\hat{\alpha} + \hat{\beta}_1(x + 1)\} = \hat{\beta}_2 + \hat{\beta}_3(x + 1)$. Thus, the estimated coefficient for the interaction effect term $\hat{\beta}_3 = \{\hat{\beta}_2 + \hat{\beta}_3(x+1)\} - (\hat{\beta}_2 + \hat{\beta}_3 x)$ represents the estimated difference in the average treatment effect between two groups of voters whose ages differ by one year.

To compute this estimated difference in R, we first fit the linear regression model with the interaction term between the `age` and `Neighbors` variables. We use the syntax `age * messages`, which produces the main terms and the interaction term.

```
fit.age <- lm(primary2006 ~ age * messages, data = social.neighbor)
fit.age

##
## Call:
## lm(formula = primary2006 ~ age * messages, data = social.neighbor)
##
## Coefficients:
##           (Intercept)                    age
##             0.0974733              0.0039982
##     messagesNeighbors  age:messagesNeighbors
##             0.0498294              0.0006283
```

The result suggests that the estimated difference in the average treatment effect between two groups of voters whose ages differ by one year is equal to 0.06 percentage points. Based on this regression model, we can also compute the estimated average treatment effect for different ages. We choose 25, 45, 65, and 85 years old for illustration. We use the `predict()` function by providing the `newdata` argument with a data frame that contains these ages as separate observations.

```
## age = 25, 45, 65, 85 in Neighbors group
age.neighbor <- data.frame(age = seq(from = 25, to = 85, by = 20),
                           messages = "Neighbors")
## age = 25, 45, 65, 85 in Control group
age.control <- data.frame(age = seq(from = 25, to = 85, by = 20),
                          messages = "Control")
## average treatment effect for age = 25, 45, 65, 85
ate.age <- predict(fit.age, newdata = age.neighbor) -
    predict(fit.age, newdata = age.control)
ate.age

##          1          2          3          4
## 0.06553713 0.07810329 0.09066944 0.10323560
```

Researchers have found that the linearity assumption is inappropriate when modeling turnout. While people become more likely to vote as they get older, their likelihood of voting starts decreasing in their 60s or 70s. One common strategy to address this phenomenon is to model turnout as a *quadratic function* of age by including the square of age as an additional predictor. Consider the following model, which also includes

interaction terms:

$$Y = \alpha + \beta_1 \text{ age} + \beta_2 \text{ age}^2 + \beta_3 \text{ Neighbors} + \beta_4 (\text{age} \times \text{Neighbors}) + \beta_5 (\text{age}^2 \times \text{Neighbors}) + \epsilon. \quad (4.10)$$

In R, a formula can contain mathematical functions such as a square or natural logarithm using the `I()` function. For example, to include a square of the x variable in a formula, we can use the syntax `I(x^2)`. The `I()` function enables other arithmetic operations such as `I(sqrt(x))` and `I(log(x))`. We now fit the model specified in equation (4.10).

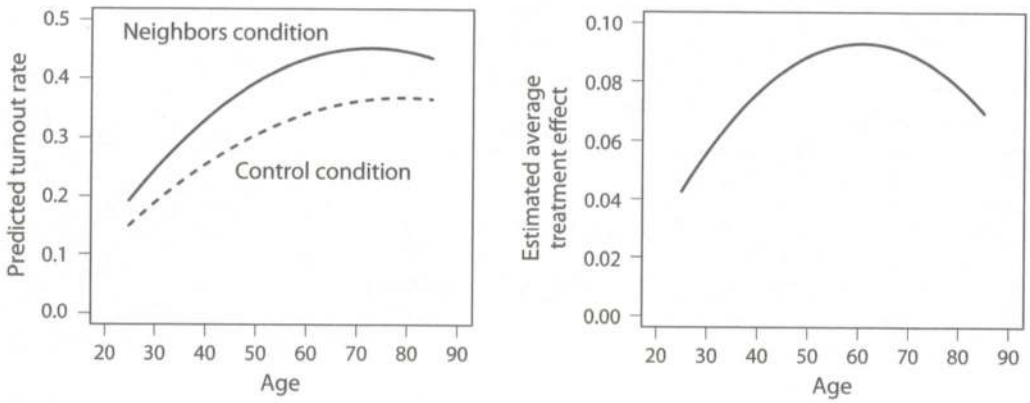
```
fit.age2 <- lm(primary2006 ~ age + I(age^2) + messages + age:messages +
                 I(age^2):messages, data = social.neighbor)
fit.age2

##
## Call:
## lm(formula = primary2006 ~ age + I(age^2) + messages + age:messages +
##     I(age^2):messages, data = social.neighbor)
##
## Coefficients:
##               (Intercept)                        age
##                 -7.385e-02                  1.143e-02
##                  I(age^2)          messagesNeighbors
##                 -7.389e-05                 -4.330e-02
##    age:messagesNeighbors  I(age^2):messagesNeighbors
##                  4.646e-03                 -3.961e-05
```

In a complicated model like this one, the coefficients no longer have an easy interpretation. In such situations, it is best to predict the average outcome under various scenarios using the `predict()` function and then compute quantities of interest. Here, we predict the average turnout rate for voters of different ages, ranging from 25 to 85, under the `Neighbors` and `Control` conditions. We then compute the average treatment effect as the difference between the two conditions and characterize it as a function of age. The following syntax accomplishes this task.

```
## predicted turnout rate under the Neighbors treatment condition
yT.hat <- predict(fit.age2,
                  newdata = data.frame(age = 25:85, messages = "Neighbors"))
## predicted turnout rate under the Control condition
yC.hat <- predict(fit.age2,
                  newdata = data.frame(age = 25:85, messages = "Control"))
```

For ease of interpretation, we plot the results. The first plot displays the predicted turnout as a function of age separately for the `Neighbors` and `Control` groups. The second plot shows the estimated average treatment effect as a function of age.

```
## plotting the predicted turnout rate under each condition
plot(x = 25:85, y = yT.hat, type = "l", xlim = c(20, 90), ylim = c(0, 0.5),
     xlab = "Age", ylab = "Predicted turnout rate")
lines(x = 25:85, y = yC.hat, lty = "dashed")
text(40, 0.45, "Neighbors condition")
text(45, 0.15, "Control condition")
## plotting the average treatment effect as a function of age
plot(x = 25:85, y = yT.hat - yC.hat, type = "l", xlim = c(20, 90),
     ylim = c(0, 0.1), xlab = "Age",
     ylab = "Estimated average\n treatment effect")
```



We find that according to this model, the estimated average treatment effect peaks around 60 years old, and the effect size is much smaller among young and old voters.

4.3.4 REGRESSION DISCONTINUITY DESIGN

The discussion in chapter 2 implies that we can interpret the association between treatment and outcome variables as causal if there is no confounding variable. This was the case in the experimental studies we analyzed in sections 4.3.1–4.3.3. In observational studies, however, the treatment assignment is not randomized. As a result, confounding factors, rather than the treatment variable, may explain the outcome difference between the treatment and control groups. In section 2.5, we discussed several research design strategies to address this potential selection bias problem. Here, we introduce another research design for observational studies called *regression discontinuity design* (RD design).

Table 4.8. Members of the British Parliament Personal Wealth Data

Variable	*Description*
`surname`	surname of the candidate
`firstname`	first name of the candidate
`party`	party of the candidate (`labour` or `tory`)
`ln.gross`	log gross wealth at the time of death
`ln.net`	log net wealth at the time of death
`yob`	year of birth of the candidate
`yod`	year of death of the candidate
`margin.pre`	margin of the candidate's party in the previous election
`region`	electoral region
`margin`	margin of victory (vote share)

As an application of RD design, we consider how much politicians can increase their personal wealth due to holding office. Scholars investigated this question by analyzing members of Parliament (MPs) in the United Kingdom.[6] The authors of the original study collected information about personal wealth at the time of death for several hundred competitive candidates who ran for office in general elections between 1950 and 1970. The data are contained in the CSV file `MPs.csv`. The names and descriptions of the variables in this data set appear in table 4.8.

A naive comparison of MPs and non-MPs in terms of their wealth is unlikely to yield valid causal inference because those who became MPs differ from those who did not in terms of many observable and unobservable characteristics. Instead, the key intuition behind RD design is to compare those candidates who narrowly won office with those who barely lost it. The idea is that when one's margin of victory switches from a negative number to a positive number, we would expect a large, discontinuous, positive jump in the personal wealth of electoral candidates if serving in office actually financially benefits them. Assuming that nothing else is going on at this point of discontinuity, we can identify the average causal effect of being an MP at this threshold by comparing the candidates who barely won the election with those who barely lost it. Regression is used to predict the average personal wealth at the point of discontinuity.

A simple scatter plot with regression lines is the best way to understand RD design. To do this, we plot the outcome variable, log net wealth at the time of death, against the margin of victory. We take the natural logarithmic transformation of wealth because this variable is quite skewed by a small number of politicians accumulating a large amount of wealth (see the discussion in section 3.4.1). We then separately fit a linear regression model to the observations with a positive margin (i.e., the candidates who won elections and became MPs) and another regression model to those with a negative margin (the candidates who lost). The difference in predicted values at the point of

[6] This application is based on Andrew C. Eggers and Jens Hainmueller (2009) "MPs for sale? Returns to office in postwar British politics." *American Political Science Review*, vol. 103, no. 4, pp. 513–533.

discontinuity, i.e., a zero margin of victory, between the two regressions represents the average causal effect on personal wealth of serving as an MP.

We begin by subsetting the data based on party (Labour and Tory) and then fit two regressions for each data set.

```
## load the data and subset them into two parties
MPs <- read.csv("MPs.csv")
MPs.labour <- subset(MPs, subset = (party == "labour"))
MPs.tory <- subset(MPs, subset = (party == "tory"))
## two regressions for Labour: negative and positive margin
labour.fit1 <- lm(ln.net ~ margin,
                  data = MPs.labour[MPs.labour$margin < 0, ])
labour.fit2 <- lm(ln.net ~ margin,
                  data = MPs.labour[MPs.labour$margin > 0, ])
## two regressions for Tory: negative and positive margin
tory.fit1 <- lm(ln.net ~ margin, data = MPs.tory[MPs.tory$margin < 0, ])
tory.fit2 <- lm(ln.net ~ margin, data = MPs.tory[MPs.tory$margin > 0, ])
```

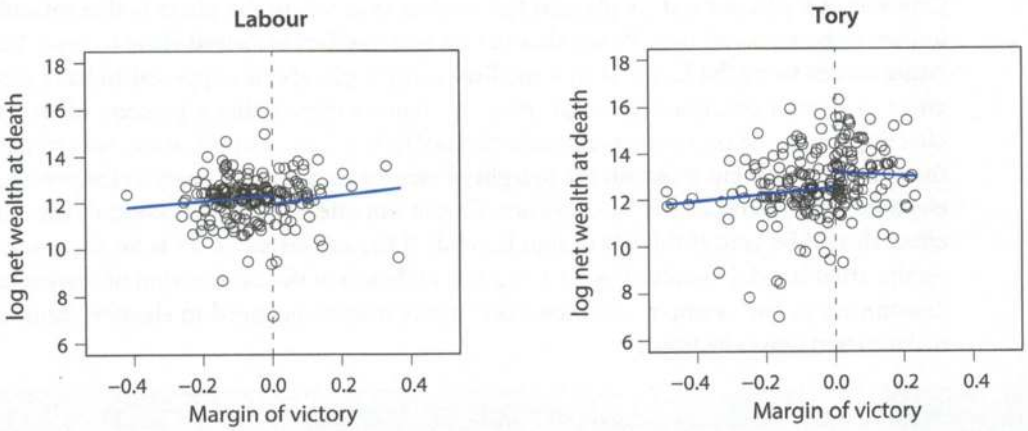
To predict the outcome using a specific value of predictor, we can use the `predict()` function by specifying a new data frame, `newdata`, as the argument. We conduct a separate analysis for Labour and Tory candidates to estimate each party's causal effect of interest.

```
## Labour: range of predictions
y1l.range <- c(min(MPs.labour$margin), 0) # min to 0
y2l.range <- c(0, max(MPs.labour$margin)) # 0 to max
## prediction
y1.labour <- predict(labour.fit1, newdata = data.frame(margin = y1l.range))
y2.labour <- predict(labour.fit2, newdata = data.frame(margin = y2l.range))
## Tory: range of predictions
y1t.range <- c(min(MPs.tory$margin), 0) # min to 0
y2t.range <- c(0, max(MPs.tory$margin)) # 0 to max
## predict outcome
y1.tory <- predict(tory.fit1, newdata = data.frame(margin = y1t.range))
y2.tory <- predict(tory.fit2, newdata = data.frame(margin = y2t.range))
```

We can now plot the predicted values for each party in the scatter plot of log net wealth and electoral margin.

```
## scatter plot with regression lines for Labour
plot(MPs.labour$margin, MPs.labour$ln.net, main = "Labour",
     xlim = c(-0.5, 0.5), ylim = c(6, 18), xlab = "Margin of victory",
     ylab = "log net wealth at death")
```

```
abline(v = 0, lty = "dashed")
## add regression lines
lines(y1l.range, y1.labour, col = "blue")
lines(y2l.range, y2.labour, col = "blue")
## scatter plot with regression lines for Tory
plot(MPs.tory$margin, MPs.tory$ln.net, main = "Tory", xlim = c(-0.5, 0.5),
     ylim = c(6, 18), xlab = "Margin of victory",
     ylab = "log net wealth at death")
abline(v = 0, lty = "dashed")
## add regression lines
lines(y1t.range, y1.tory, col = "blue")
lines(y2t.range, y2.tory, col = "blue")
```



The result suggests that Tory MPs financially benefit from serving in office whereas Labour MPs do not. How large is the effect for Tory candidates? We can numerically compute the differences in prediction at the zero margin and put them back on the original scale (pounds) since net wealth is measured on a log scale. Recall from section 3.4.1 that the *inverse function* of the natural logarithm is the exponential function, given by `exp()` in R.

```
## average net wealth for Tory MP
tory.MP <- exp(y2.tory[1])
tory.MP

##        1
## 533813.5

## average net wealth for Tory non-MP
tory.nonMP <- exp(y1.tory[2])
```

```
tory.nonMP
##         2
## 278762.5
## causal effect in pounds
tory.MP - tory.nonMP
##         1
## 255050.9
```

The estimated effect of being an MP on the personal wealth of Tory candidates is a little above 250,000 pounds. Since the average net wealth for Tory non-MPs is predicted to be a little above 270,000 pounds, the estimated effect is quite substantial. Being an MP almost doubles one's net wealth at death.

How should one examine the *internal validity* of regression discontinuity design? One way is a *placebo test*. A placebo test finds a case where the effect is theoretically known to be zero and then shows that the estimated effect is indeed close to zero. The name comes from the fact that in a medical study a placebo is supposed to have zero effect on health outcomes (though much evidence suggests that a placebo often has effects, perhaps via psychological mechanisms). In the current application, we estimate the average treatment effect on the margin of victory for the same party in the *previous* election. Since being an MP in the future should not affect the past election result, this effect should be zero if the RD design is valid. If the estimated effect is far from zero, on the other hand, it would suggest a possible violation of the assumption of regression discontinuity. For example, the incumbent party may be engaged in election fraud in order to win close elections.

```
## two regressions for Tory: negative and positive margin
tory.fit3 <- lm(margin.pre ~ margin, data = MPs.tory[MPs.tory$margin < 0, ])
tory.fit4 <- lm(margin.pre ~ margin, data = MPs.tory[MPs.tory$margin > 0, ])
## the difference between two intercepts is the estimated effect
coef(tory.fit4)[1] - coef(tory.fit3)[1]
## (Intercept)
## -0.01725578
```

The estimated effect on the previous margin of victory is less than 2 percentage points. This small effect size gives empirical support for the claim that RD design is applicable to this study. In chapter 7, we will more formally answer the question of how small is small enough to reach this conclusion.

While RD design can overcome the main difficulty of observational studies, i.e., potential confounding bias, this strength of *internal validity* comes at the cost of *external validity*. Specifically, the estimated causal effects obtained under this design apply only to the observations near the point of discontinuity. In our application, these observations represent candidates who narrowly won or lost elections. The degree to which MPs benefit financially from serving in office may be quite different for those

who won elections by a larger margin. Thus, although RD requires weaker assumptions than other approaches, the resulting estimates may not be generalizable to a larger population of interest.

Regression discontinuity design (RD design) is a research design strategy for causal inference in observational studies with possible confounding factors. RD design assumes that the change in outcome at the point of discontinuity can be attributed to the change in the treatment variable alone. While RD design often has strong internal validity, it may lack external validity because the result may not be generalizable to observations away from the point of discontinuity.

4.4 Summary

We began this chapter with a discussion of election forecasting. We showed that preelection polls can be used to obtain relatively accurate, though not perfect, **predictions** of election outcomes in the context of US presidential elections. We introduced **prediction error** and explained how the accuracy of prediction can be measured using statistics such as **bias** and the **root-mean-squared error**. We also discussed the problem of **classification**, which is the prediction of categorical outcomes. Two types of **misclassification** are possible—false positives and false negatives. For example, a voter who did turn out being classified as a nonvoter would be a false negative, whereas a voter who did not turn out being classified as a voter would be a false positive. There is a clear trade-off between the two: minimizing false positives tends to increase false negatives and vice versa.

We then introduced a **linear regression model** as a commonly used method to predict an outcome variable of interest using another variable. The model enables researchers to predict an outcome variable based on the values of explanatory variables or predictors. Predictions based on the linear regression model are typically obtained through the **method of least squares** by minimizing the sum of squared prediction errors. We discussed the exact relationship between linear regression and **correlation**, and the phenomenon called **regression towards the mean**. Finally, we presented several ways to assess model fit through the examination of the **coefficient of determination** and residuals. It is important to avoid overfitting one's model to the data at hand so that the model does not capture any idiosyncratic characteristics of the sample and instead identifies the systematic features of the data-generating process.

Despite our intuition, association discovered through regression does not necessarily imply **causation**. A regression's ability to predict observable outcomes does not necessarily entail ability to predict counterfactual outcomes. Yet, valid causal inference requires the latter. At the end of the chapter, we discussed the use of regression in the analysis of experimental and observational data. We discussed how to estimate heterogeneous treatment effects using the linear regression model with **interaction terms**. We also discussed the **regression discontinuity design**. By exploiting the discontinuity in the treatment assignment mechanism, this design enables researchers

Table 4.9. Intrade Prediction Market Data from 2008 and 2012.

Variable	*Description*
`day`	date of the session
`statename`	full name of each state (including District of Columbia in 2008)
`state`	abbreviation of each state (including District of Columbia in 2008)
`PriceD`	closing price (predicted vote share) of the Democratic nominee's market
`PriceR`	closing price (predicted vote share) of the Republican nominee's market
`VolumeD`	total session trades of the Democratic Party nominee's market
`VolumeR`	total session trades of the Republican Party nominee's market

to credibly identify causal effects in observational studies. The main disadvantage of the regression discontinuity design, however, is the potential lack of **external validity**. Specifically, the empirical conclusions based on this design may not be applicable beyond the observations close to the discontinuity threshold.

4.5 Exercises

4.5.1 PREDICTION BASED ON BETTING MARKETS

Earlier in the chapter, we studied the prediction of election outcomes using polls. Here, we study the prediction of election outcomes based on betting markets. In particular, we analyze data for the 2008 and 2012 US presidential elections from the online betting company called Intrade. At Intrade, people trade contracts such as "Obama to win the electoral votes of Florida." Each contract's market price fluctuates based on its sales. Why might we expect betting markets like Intrade to accurately predict the outcomes of elections or of other events? Some argue that the market can aggregate available information efficiently. In this exercise, we will test this *efficient market hypothesis* by analyzing the market prices of contracts for Democratic and Republican nominees' victories in each state.

The data files for 2008 and 2012 are available in CSV format as `intrade08.csv` and `intrade12.csv`, respectively. Table 4.9 presents the names and descriptions of these data sets. Each row of the data sets represents daily trading information about the contracts for either the Democratic or Republican Party nominee's victory in a particular state. We will also use the election outcome data. These data files are `pres08.csv` (table 4.1) and `pres12.csv` (table 4.5).

1. We will begin by using the market prices on the day before the election to predict the 2008 election outcome. To do this, subset the data such that it contains the market information for each state and candidate on the day before the election only. Note that in 2008, Election Day was November 4. We compare the closing prices for the two candidates in a given state and classify a candidate whose contract has a higher price as the predicted winner of that state. Which states

were misclassified? How does this compare to the classification by polls presented earlier in this chapter? Repeat the same analysis for the 2012 election, which was held on November 6. How well did the prediction market do in 2012 compared to 2008? Note that in 2012 some less competitive states have missing data on the day before the election because there were no trades on the Republican and Democratic betting markets. Assume Intrade predictions would have been accurate for these states.

2. How do the predictions based on the betting markets change over time? Implement the same classification procedure as above on each of the last 90 days of the 2008 campaign rather than just the day before the election. Plot the predicted number of electoral votes for the Democratic Party nominee over this 90-day period. The resulting plot should also indicate the actual election result. Note that in 2008, Obama won 365 electoral votes. Briefly comment on the plot.

3. Repeat the previous exercise but this time use the seven-day *moving-average* price, instead of the daily price, for each candidate within a state. Just as in section 4.1.3, this can be done with a loop. For a given day, we take the average of the Session Close prices within the past seven days (including that day). To answer this question, we must first compute the seven-day average within each state. Next, we sum the electoral votes for the states Obama is predicted to win. Using the `tapply()` function will allow us to efficiently compute the predicted winner for each state on a given day.

4. Create a similar plot for 2008 statewide poll predictions using the data file `polls08.csv` (see table 4.2). Notice that polls are not conducted daily within each state. Therefore, within a given state, for each of the last 90 days of the campaign, we compute the average margin of victory from the most recent poll(s) conducted. If multiple polls occurred on the same day, average these polls. Based on the most recent predictions in each state, sum Obama's total number of predicted electoral votes. One strategy to answer this question is to program two loops—an inner loop with 51 iterations (for each state) and an outer loop with 90 iterations (for each day).

5. What is the relationship between the price margins of the Intrade market and the actual margin of victory? Using the market data from the day before the election in 2008 only, regress Obama's actual margin of victory in each state on Obama's price margin from the Intrade markets. Similarly, in a separate analysis, regress Obama's actual margin of victory on Obama's predicted margin from the latest polls within each state. Interpret the results of these regressions.

6. Do the 2008 predictions of polls and Intrade accurately predict each state's 2012 elections results? Using the fitted regressions from the previous question, forecast Obama's actual margin of victory for the 2012 election in two ways. First, use the 2012 Intrade price margins from the day before the election as the predictor in each state. Recall that the 2012 Intrade data do not contain market prices for all

Table 4.10. 2012 US Presidential Election Polling Data.

Variable	*Description*
`state`	abbreviated name of the state in which the poll was conducted
`Obama`	predicted support for Obama (percentage)
`Romney`	predicted support for Romney (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted

states. Ignore states without data. Second, use the 2012 poll-predicted margins from the latest polls in each state as the predictor, found in `polls12.csv`. Table 4.10 presents the names and descriptions of the 2012 US presidential election polling data.

4.5.2 ELECTION AND CONDITIONAL CASH TRANSFER PROGRAM IN MEXICO

In this exercise, we analyze the data from a study that seeks to estimate the electoral impact of *Progresa*, Mexico's *conditional cash transfer program* (CCT program).[7] The original study relied on a randomized evaluation of the CCT program in which eligible villages were randomly assigned to receive the program either 21 months (early Progresa) or 6 months (late Progresa) before the 2000 Mexican presidential election. The author of the original study hypothesized that the CCT program would mobilize voters, leading to an increase in turnout and support for the incumbent party (PRI, or Partido Revolucionario Institucional, in this case). The analysis was based on a sample of precincts that contain at most one participating village in the evaluation.

The data we analyze are available as the CSV file `progresa.csv`. Table 4.11 presents the names and descriptions of variables in the data set. Each observation in the data represents a precinct, and for each precinct the file contains information about its treatment status, the outcomes of interest, socioeconomic indicators, and other precinct characteristics.

1. Estimate the impact of the CCT program on turnout and support for the incumbent party (PRI) by comparing the average electoral outcomes in the "treated" (early Progresa) precincts versus the ones observed in the "control" (late Progresa) precincts. Next, estimate these effects by regressing the outcome variable on the treatment variable. Interpret and compare the estimates under these approaches. Here, following the original analysis, use the turnout and support rates as shares of the eligible voting population (`t2000` and `pri2000s`, respectively). Do the results support the hypothesis? Provide a brief interpretation.

[7] This exercise is based on the following articles: Ana de la O (2013) "Do conditional cash transfers affect voting behavior? Evidence from a randomized experiment in Mexico." *American Journal of Political Science*, vol. 57, no. 1, pp. 1–14 and Kosuke Imai, Gary King, and Carlos Velasco (2015) "Do nonpartisan programmatic policies have partisan electoral effects? Evidence from two large scale randomized experiments." Working paper.

Table 4.11. Conditional Cash Transfer Program (Progresa) Data.

Variable	*Description*
`treatment`	whether an electoral precinct contains a village where households received early Progresa
`pri2000s`	PRI votes in the 2000 election as a share of precinct population above 18
`pri2000v`	official PRI vote share in the 2000 election
`t2000`	turnout in the 2000 election as a share of precinct population above 18
`t2000r`	official turnout in the 2000 election
`pri1994`	total PRI votes in the 1994 presidential election
`pan1994`	total PAN votes in the 1994 presidential election
`prd1994`	total PRD votes in the 1994 presidential election
`pri1994s`	total PRI votes in the 1994 election as a share of precinct population above 18
`pan1994s`	total PAN votes in the 1994 election as a share of precinct population above 18
`prd1994s`	total PRD votes in the 1994 election as a share of precinct population above 18
`pri1994v`	official PRI vote share in the 1994 election
`pan1994v`	official PAN vote share in the 1994 election
`prd1994v`	official PRD vote share in the 1994 election
`t1994`	turnout in the 1994 election as a share of precinct population above 18
`t1994r`	official turnout in the 1994 election
`votos1994`	total votes cast in the 1994 presidential election
`avgpoverty`	precinct average of village poverty index
`pobtot1994`	total population in the precinct
`villages`	number of villages in the precinct

2. In the original analysis, the author fits a linear regression model that includes, as predictors, a set of pretreatment covariates as well as the treatment variable. Here, we fit a similar model for each outcome that includes the average poverty level in a precinct (`avgpoverty`), the total precinct population in 1994 (`pobtot1994`), the total number of voters who turned out in the previous election (`votos1994`), and the total number of votes cast for each of the three main competing parties in the previous election (`pri1994` for PRI, `pan1994` for Partido Acción Nacional or PAN, and `prd1994` for Partido de la Revolución Democrática or PRD). Use the same outcome variables as in the original analysis, which are based on the shares of the voting age population. According to this model, what are the estimated average effects of the program's availability on turnout and support for the incumbent party? Are these results different from those you obtained in the previous question?

3. Next, we consider an alternative, and more natural, model specification. We will use the original outcome variables as in the previous question. However, our model should include the previous election outcome variables measured as shares of the voting age population (as done for the outcome variables `t1994`, `pri1994s`, `pan1994s`, and `prd1994s`) instead of those measured in counts. In addition, we apply the natural logarithmic transformation to the precinct population variable when including it as a predictor. As in the original model, our model includes the average poverty index as an additional predictor. Are the results based on these new model specifications different from those we obtained in the previous question? If the results are different, which model fits the data better?

4. We examine the balance of some pretreatment variables used in the previous analyses. Using box plots, compare the distributions of the precinct population (on the original scale), average poverty index, previous turnout rate (as a share of the voting age population), and previous PRI support rate (as a share of the voting age population) between the treatment and control groups. Comment on the patterns you observe.

5. We next use the official turnout rate `t2000r` (as a share of the registered voters) as the outcome variable rather than the turnout rate used in the original analysis (as a share of the voting age population). Similarly, we use the official PRI's vote share `pri2000v` (as a share of all votes cast) rather than the PRI's support rate (as a share of the voting age population). Compute the average treatment effect of the CCT program using a linear regression with the average poverty index, the log-transformed precinct population, and the previous official election outcome variables (`t1994r` for the previous turnout; `pri1994v`, `pan1994v`, and `prd1994v` for the previous PRI, PAN, and PRD vote shares). Briefly interpret the results.

6. So far we have focused on estimating the average treatment effects of the CCT program. However, these effects may vary from one precinct to another. One important dimension to consider is poverty. We may hypothesize that since individuals in precincts with higher levels of poverty are more receptive to cash transfers, they are more likely to turn out in the election and support the incumbent party when receiving the CCT program. Assess this possibility by examining how the average treatment effect of the policy varies by different levels of poverty for precincts. To do so, fit a linear regression with the following predictors: the treatment variable, the log-transformed precinct population, the average poverty index and its square, the interaction between the treatment and the poverty index, and the interaction between the treatment and the squared poverty index. Estimate the average effects for unique observed values and plot them as a function of the average poverty level. Comment on the resulting plot.

Table 4.12. Brazilian Government Transfer Data.

Variable	*Description*
`pop82`	population in 1982
`poverty80`	poverty rate of the state in 1980
`poverty91`	poverty rate of the state in 1991
`educ80`	average years in education of the state in 1980
`educ91`	average years in education of the state in 1991
`literate91`	literacy rate of the state in 1991
`state`	state
`region`	region
`id`	municipal ID
`year`	year of measurement

4.5.3 GOVERNMENT TRANSFER AND POVERTY REDUCTION IN BRAZIL

In this exercise, we estimate the effects of increased government spending on educational attainment, literacy, and poverty rates.[8] Some scholars argue that government spending accomplishes very little in environments of high corruption and inequality. Others suggest that in such environments, accountability pressures and the large demand for public goods will drive elites to respond. To address this debate, we exploit the fact that until 1991, the formula for government transfers to individual Brazilian municipalities was determined in part by the municipality's population. This meant that municipalities with populations below the official cutoff did not receive additional revenue, while states above the cutoff did. The data set `transfer.csv` contains the variables shown in table 4.12.

1. We will apply the regression discontinuity design to this application. State the required assumption for this design and interpret it in the context of this specific application. What would be a scenario in which this assumption is violated? What are the advantages and disadvantages of this design for this specific application?

2. Begin by creating a variable that determines how close each municipality was to the cutoff that determined whether states received a transfer or not. Transfers occurred at three separate population cutoffs: 10,188, 13,584, and 16,980. Using these cutoffs, create a single variable that characterizes the difference from the *closest* population cutoff. Following the original analysis, standardize this measure by dividing the difference by the corresponding cutoff, and multiplying it by 100. This will yield a normalized percentage score for the difference between the population of each state and the cutoff, relative to the cutoff value.

[8] This exercise is based on Stephan Litschig and Kevin M. Morrison (2013) "The impact of intergovernmental transfers on education outcomes and poverty reduction." *American Economic Journal: Applied Economics*, vol. 5, no. 4, pp. 206–240.

3. Begin by subsetting the data to include only those municipalities within 3 points of the funding cutoff on either side. Using regressions, estimate the average causal effect of government transfer on each of the three outcome variables of interest: educational attainment, literacy, and poverty. Give a brief substantive interpretation of the results.

4. Visualize the analysis performed in the previous question by plotting data points, fitted regression lines, and the population threshold. Briefly comment on the plot.

5. Instead of fitting linear regression models, we compute the difference-in-means of the outcome variables between the groups of observations above the threshold and below it. How do the estimates differ from what you obtained in question 3? Is the assumption invoked here identical to the one required for the analysis conducted in question 3? Which estimates are more appropriate? Discuss.

6. Repeat the analysis conducted in question 3 but vary the width of the analysis window from 1 to 5 percentage points below and above the threshold. Obtain the estimate for every percentage point. Briefly comment on the results.

7. Conduct the same analysis as in question 3 but this time using the measures of poverty rate and educational attainment taken in 1980, before the population-based government transfers began. What do the results suggest about the validity of the analysis presented in question 3?