

# **Quantitative Social Science**

---



# **Quantitative Social Science**

---

An Introduction in Stata

KOSUKE IMAI  
LORI D. BOUGHER

PRINCETON UNIVERSITY PRESS  
**Princeton and Oxford**

Copyright © 2021 by Princeton University Press

Princeton University Press is committed to the protection of copyright and the intellectual property our authors entrust to us. Copyright promotes the progress and integrity of knowledge. Thank you for supporting free speech and the global exchange of ideas by purchasing an authorized edition of this book. If you wish to reproduce or distribute any part of it in any form, please obtain permission.

Requests for permission to reproduce material from this work  
should be sent to [permissions@press.princeton.edu](mailto:permissions@press.princeton.edu)

Published by Princeton University Press  
41 William Street, Princeton, New Jersey 08540  
6 Oxford Street, Woodstock, Oxfordshire OX20 1TR

[press.princeton.edu](http://press.princeton.edu)

All Rights Reserved

Library of Congress Cataloging-in-Publication Data

Names: Imai, Kosuke, author. | Bouger, Lori D., 1979– author.

Title: Quantitative social science : an introduction in Stata / Kosuke Imai, Lori D. Bouger.

Description: Princeton : Princeton University Press, [2021] | Includes bibliographical references and index.

Identifiers: LCCN 2020040182 (print) | LCCN 2020040183 (ebook) | ISBN 9780691191096 (paperback ;

alk. paper) | ISBN 9780691191089 (hardback ; alk. paper) | ISBN 9780691191294 (ebook)

Subjects: LCSH: Social sciences—Methodology. | Social sciences—Research. | Quantitative research—  
Data processing. | Stata.

Classification: LCC H62 .I5365 2021 (print) | LCC H62 (ebook) | DDC 300.72/1—dc23

LC record available at <https://lccn.loc.gov/2020040182> (<https://lccn.loc.gov/2020040182>)

LC ebook record available at <https://lccn.loc.gov/2020040183> (<https://lccn.loc.gov/2020040183>)

British Library Cataloging-in-Publication Data is available

Editorial: Bridget Flannery-McCoy and Alena Chekanov

Production Editorial: Mark Bellis

Text Design: C. Alvarez-Gaffin

Cover Design: Wanda España

Production: Erin Suydam

Publicity: Kate Hensley and Kate Farquhar-Thomson

Copyeditor: Theresa Kornak

This book has been composed in  $\text{\LaTeX}$

Printed on acid-free paper.  $\infty$

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To our families

Christina, Keiji, and Misaki

Ken, Sandy, and Brandon



# Contents

---

List of Tables	xiii
List of Figures	xv
Preface	xvii
Preface to the Original Book	xix
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Overview of the Book	3
1.2 How to Use this Book	7
1.3 Introduction to Stata	8
1.3.1 Arithmetic Operations	9
1.3.2 Variables	10
1.3.3 Labels	16
1.3.4 Describing the Data	18
1.3.5 Data Files	20
1.3.6 Merging Data Sets in Stata	21
1.3.7 Packages	23
1.3.8 Programming and Learning Tips	25
1.4 Summary	26
1.5 Exercises	27
1.5.1 Bias in Self-Reported Turnout	27
1.5.2 Understanding World Population Dynamics	28
<b>2 CAUSALITY</b>	<b>32</b>
2.1 Racial Discrimination in the Labor Market	32
2.2 Subsetting the Data in Stata	38
2.2.1 Relational Operators	38
2.2.2 Logical Operators	39
2.2.3 Simple Conditional Statements and Variable Creation	40
2.2.4 Subsetting Using Conditions	43
2.2.5 Preserving and Transforming Data Sets	44
2.3 Causal Effects and the Counterfactual	47

2.4 Randomized Controlled Trials	49
2.4.1 The Role of Randomization	50
2.4.2 Social Pressure and Voter Turnout	51
2.5 Observational Studies	56
2.5.1 Minimum Wage and Unemployment	56
2.5.2 Confounding Bias	60
2.5.3 Before-and-After and Difference-in-Differences Designs	63
2.6 Descriptive Statistics for a Single Variable	67
2.6.1 Quantiles	67
2.6.2 Standard Deviation	72
2.7 Summary	75
2.8 Exercises	75
2.8.1 Efficacy of Small Class Size in Early Education	75
2.8.2 Changing Minds on Gay Marriage	77
2.8.3 Success of Leader Assassination as a Natural Experiment	79
<b>3 MEASUREMENT</b>	<b>81</b>
3.1 Measuring Civilian Victimization during Wartime	81
3.2 Handling Missing Data in Stata	84
3.2.1 Missings Package	85
3.3 Visualizing the Univariate Distribution	86
3.3.1 Bar Plot	87
3.3.2 Histogram	89
3.3.3 Box Plot	92
3.3.4 Printing and Saving Graphs	94
3.4 Survey Sampling	96
3.4.1 The Role of Randomization	96
3.4.2 Nonresponse and Other Sources of Bias	101
3.5 Measuring Political Polarization	105
3.6 Summarizing Bivariate Relationships	106
3.6.1 Scatterplot	106
3.6.2 Correlation	109
3.6.3 Quantile–Quantile Plot	114
3.7 Clustering	117
3.7.1 The $k$ -Means Algorithm	117
3.8 Summary	121
3.9 Exercises	122
3.9.1 Changing Minds on Gay Marriage: Revisited	122
3.9.2 Political Efficacy in China and Mexico	123
3.9.3 Voting in the United Nations General Assembly	125
<b>4 PREDICTION</b>	<b>128</b>
4.1 Predicting Election Outcomes	128
4.1.1 Macros	129
4.1.2 Loops	131
4.1.3 Poll Predictions	133

4.2 Linear Regression	144
4.2.1 Facial Appearance and Election Outcomes	144
4.2.2 Correlation and Scatterplots	146
4.2.3 Least Squares	148
4.2.4 Regression toward the Mean	154
4.2.5 Model Fit	160
4.3 Regression and Causation	167
4.3.1 Randomized Experiments	167
4.3.2 Regression with Multiple Predictors	171
4.3.3 Heterogeneous Treatment Effects	177
4.3.4 Regression Discontinuity Design	184
4.4 Summary	190
4.5 Exercises	190
4.5.1 Prediction Based on Betting Markets	190
4.5.2 Election and Conditional Cash Transfer Program in Mexico	193
4.5.3 Government Transfer and Poverty Reduction in Brazil	195
<b>5 PROBABILITY</b>	<b>197</b>
5.1 Probability	197
5.1.1 Frequentist versus Bayesian	197
5.1.2 Definition and Axioms	199
5.1.3 Permutations	202
5.1.4 Sampling with and without Replacement	205
5.1.5 Combinations	208
5.2 Conditional Probability	210
5.2.1 Conditional, Marginal, and Joint Probabilities	210
5.2.2 Independence	218
5.2.3 Bayes' Rule	226
5.2.4 Predicting Race Using Surname and Residence Location	228
5.3 Random Variables and Probability Distributions	241
5.3.1 Random Variables	241
5.3.2 Bernoulli and Uniform Distributions	241
5.3.3 Binomial Distribution	246
5.3.4 Normal Distribution	249
5.3.5 Expectation and Variance	256
5.3.6 Predicting Election Outcomes with Uncertainty	260
5.4 Large Sample Theorems	264
5.4.1 The Law of Large Numbers	264
5.4.2 The Central Limit Theorem	266
5.5 Summary	271
5.6 Exercises	272
5.6.1 The Mathematics of Enigma	272
5.6.2 A Probability Model for Betting Market Election Prediction	274

<b>6 UNCERTAINTY</b>	<b>276</b>
6.1 Estimation	276
6.1.1 Unbiasedness and Consistency	277
6.1.2 Standard Error	285
6.1.3 Confidence Intervals	290
6.1.4 Margin of Error and Sample Size Calculation in Polls	296
6.1.5 Analysis of Randomized Controlled Trials	301
6.1.6 Analysis Based on Student's <i>t</i> -Distribution	304
6.2 Hypothesis Testing	307
6.2.1 Tea-Tasting Experiment	307
6.2.2 The General Framework	313
6.2.3 One-Sample Tests	316
6.2.4 Two-Sample Tests	322
6.2.5 Pitfalls of Hypothesis Testing	327
6.2.6 Power Analysis	329
6.3 Linear Regression Model with Uncertainty	336
6.3.1 Linear Regression as a Generative Model	337
6.3.2 Unbiasedness of Estimated Coefficients	343
6.3.3 Standard Errors of Estimated Coefficients	345
6.3.4 Inference about Coefficients	348
6.3.5 Inference about Predictions	350
6.4 Summary	356
6.5 Exercises	357
6.5.1 Sex Ratio and the Price of Agricultural Crops in China	357
6.5.2 Filedrawer and Publication Bias in Academic Research	359
6.5.3 The 1932 German Election in the Weimar Republic	361
<b>7 DISCOVERY</b>	<b>364</b>
7.1 Network Data	364
7.1.1 Marriage Network in Renaissance Florence	365
7.1.2 Undirected Graph and Centrality Measures	367
7.1.3 Twitter Following Network	375
7.1.4 Directed Graph and Centrality	376
7.2 Spatial Data	386
7.2.1 The 1854 Cholera Outbreak in London	386
7.2.2 Spatial Data in Stata	389
7.2.3 United States Presidential Elections	393
7.2.4 Expansion of Walmart	395
7.2.5 Animation in Stata	397
7.3 Textual Data	400
7.3.1 The Disputed Authorship of <i>The Federalist Papers</i>	400
7.3.2 Topic Discovery	404
7.3.3 Document-Term Matrix and Clusters	411
7.3.4 Authorship Prediction	413
7.3.5 Cross Validation	417

7.4 Summary	420
7.5 Exercises	420
7.5.1 International Trade Network	420
7.5.2 Mapping US Presidential Election Results over Time	422
7.5.3 Analyzing the Preambles of Constitutions	424
<b>8 NEXT</b>	<b>429</b>
General Index	433
Stata Index	439
Stata Command Abbreviation List	443



## List of Tables

---

1.1	World population estimates	11
1.2	US election turnout data	28
1.3	Fertility and mortality estimates data	29
2.1	Résumé experiment data	33
2.2	Logical conjunction and disjunction	40
2.3	Potential outcome framework of causal inference	48
2.4	Social pressure experiment data	53
2.5	Minimum-wage study data	57
2.6	STAR project data	76
2.7	Gay marriage data	78
2.8	Leader assassination data	79
3.1	Afghanistan survey data	82
3.2	Afghanistan village data	99
3.3	Legislative ideal points data	106
3.4	US Gini coefficient data	111
3.5	Gay marriage data (reshaped)	123
3.6	CCAP survey data	123
3.7	Vignette survey data	124
3.8	United Nations ideal point data	126
4.1	2008 US presidential election data	133
4.2	2008 US presidential election polling data	133
4.3	Confusion matrix	140
4.4	Facial appearance experiment data	145
4.5	2012 US presidential election data	156
4.6	1996 and 2000 US presidential election data for Florida counties	161
4.7	Women as policymakers data	168
4.8	Members of the British Parliament personal wealth data	185
4.9	Prediction market data	191
4.10	2012 US presidential election polling data	192
4.11	Conditional cash transfer program data	193
4.12	Brazilian government transfer data	196
5.1	Sample of Florida registered voter list	212
5.2	An example of a joint probability table	215
5.3	US Census surname list	230
5.4	Florida census data	234

6.1	Critical values	292
6.2	Tea-tasting experiment	308
6.3	Type I and type II errors	314
6.4	Chinese births and crops data	358
6.5	File drawer and publication bias data I	360
6.6	File drawer and publication bias data II	360
6.7	The 1932 German election data	362
7.1	Florence marriage network data	365
7.2	Twitter-following data	376
7.3	Walmart store opening data	396
7.4	<i>The federalist papers</i> data	402
7.5	International trade data	421
7.6	County-level US presidential elections data	422
7.7	The Constitution Preamble data	425

## List of Figures

---

1.1 Screenshot of Stata (version 14)	9
1.2 Screenshot of the Net search package results	24
1.3 Screenshot of the Stata Do-File Editor	25
2.1 Naming-and-shaming get-out-the-vote message	52
2.2 The difference-in-differences design in the minimum wage study	65
3.1 Harry Truman with the erroneous headline	98
3.2 The natural logarithm	100
3.3 Spatial voting model	105
3.4 Gini coefficient and Lorenz curve	109
4.1 Electoral College map of the 2008 US presidential election	129
4.2 Example pictures of candidates used in the experiment	145
4.3 Correlation coefficients and patterns of the data cloud in scatterplots	147
4.4 Facial appearance and vote shares	151
4.5 Galton's regression toward mediocrity	155
4.6 Butterfly ballot in Palm Beach County	165
5.1 Reverend Thomas Bayes	198
5.2 Venn diagram	202
5.3 Tree diagram for permutations	203
5.4 Schwarzenegger's veto message	209
5.5 Bernoulli distribution	243
5.6 Uniform distribution	244
5.7 Binomial distribution	247
5.8 Pascal's triangle	249
5.9 The normal distribution	250
5.10 The area of the normal distribution	252
5.11 Quincunx	267
5.12 The Enigma machine	272
6.1 Critical values based on the standard normal distribution	291
6.2 Sampling distribution for the tea-tasting experiment	309
6.3 One-sided and two-sided $p$ -values	318
6.4 The distribution of $p$ -values in published studies	328
6.5 Two animal oracles	329
6.6 Illustration of power analysis	331
7.1 Degree, closeness, and betweenness in undirected network	368
7.2 Degree, closeness, and betweenness in a directed network	377

7.3	John Snow's map of fatal cholera cases in London	387
7.4	John Snow's map of the natural experiment	388
7.5	<i>The Federalist Papers</i>	401
7.6	Cosine similarity of two vectors	426

## Preface

---

After the initial publication of *Quantitative Social Science: An Introduction* (QSS) in 2017, Princeton University Press received numerous inquiries from faculty and students about potential plans for a Stata version of the book. This was not a surprise since Stata, along with R, is widely used by social scientists. We hope that many students in social sciences who wish to learn data analysis and statistics with Stata find this book useful. Like the original version of QSS, the book shows how data analysis can be used to answer interesting and important questions in social sciences.

While “translating” QSS from R to Stata initially sounded straightforward, we quickly realized the difficulty of this task. In fact, after finally completing the project, we can confidently say that it would be much easier to translate the book into a foreign language! Because QSS embeds computer code within the discussion of analytical and statistical concepts, the change from R to Stata meant that many parts of the book needed to be rewritten. In addition, some data analyses are easier to conduct in one computer language than another. While Stata provides simple commands for many routine data analyses, the contents of the Discovery chapter proved to be particularly challenging. Therefore, we decided to move this chapter to the end of the book.

Without the support of many people, we would not have been able to complete this translated version. In particular, we are grateful to Raymond Hicks, who made numerous contributions at the beginning of the project. His vast knowledge of Stata was essential for the development of the book. At Princeton University Press, Eric Crahan was responsible for the idea of the translation and encouraged us to take up the challenge. Although we had some moments of doubt from time to time, we are proud to have completed the book. Bridget Flannery-McCoy inherited this project from Eric and has been a patient supporter. We especially appreciate her flexibility about deadlines as well as her understanding of the importance of pedagogy and presentation.

Kosuke Imai  
Cambridge, Massachusetts

Lori D. Bougher  
Princeton, New Jersey

June 2020



## Preface to the Original Book

---

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, but 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 at the forefront 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 would understand why those materials are necessary to understand and 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 logics.

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 Office of the Dean of the College 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 about how introductory data analysis and statistics should be taught. They allowed me to design a course at Freshman Scholars Institute (FSI), and many of the ideas in this book were born during FSI. The FSI is a great diversity initiative for first-generation college students and I am proud of being a

part of it. I am also grateful to the university administrators for their generous support on my teaching initiatives. They include Jill Dolan, Chris Eisgruber, Dave Lee, Nolan McCarty, Debbie Prentice, and Val Smith.

I also thank my co-instructors who helped me develop the materials included in this book. James Lo, Jonathan Olmsted, and Will Lowe made significant contributions to POL245 taught during FSI. I was also 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, Winston Chou, Elisha Cohen, Brandon de la Cuesta, Ted Enamorado, Matt Incantalupo, Tolya Levshin, 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, Katie McCabe, Grace Rehaut, and Ruby Shao. 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 also wish to thank Eric Crahan 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 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.

I would like to thank 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 the 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  
June 2016  
Princeton, New Jersey

## 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 past 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, has meant 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 grown rapidly. In the past, researchers largely relied on 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 such as 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 geolocation, 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, short message service (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 a rare approach, 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 policymakers, 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. 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 toward 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 Stata programming might also find this book useful, although here the emphasis is on how to use Stata to answer quantitative social science questions.

As mentioned previously, 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 40 data sets tied to original publications. Readers are strongly encouraged to consult these publications to not only better understand the data with which they will be working, but to also gain exposure to new techniques and real-world applications of data analysis.

The book consists of eight chapters. The current introductory chapter explains how to best utilize the book and presents a brief introduction to Stata, a popular statistical programming environment. Stata is available in several forms and runs on Macintosh, Windows, and Linux computers. The examples in this book were created using Stata 14. With the exception of the text analysis parts of chapter 7, the code contained in this book should also run on both earlier and later versions.

This chapter ends with two exercises that are designed to help readers practice elementary Stata 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://qss.princeton.press>. 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 7, the book focuses on basic syntax in Stata and does not introduce the wide range of additional packages that are available. However, upon completion of this book, readers will have acquired enough Stata 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, such as 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. Another example in chapter 2 includes 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 Stata programming, chapter 2 covers logical statements and subsetting. This chapter also reviews measures of the spread of data, including quantiles and standard deviation.

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 for 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. We present various ways to visualize univariate and bivariate data in Stata. We also introduce the concept of correlation and the Gini coefficient. 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 policymakers 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 toward 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 important programming concepts that enable easier and more efficient coding: macros and loops. The exercises at the end of the chapter include an analysis of whether betting markets can precisely forecast election outcomes.

Chapter 5 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 introduced in chapter 6. 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. 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. One exercise adds uncertainty to the forecasts of election outcomes based on betting market data. In a second exercise, we inspect the German cryptography machine from World War II (Enigma).

Chapter 6 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 5, chapter 6 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.

Chapter 7 is about the *discovery* of patterns from less traditional types of data. When analyzing "big data," we need automated methods and visualization tools to identify consistent patterns in the data. First, we 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. We then examine geospatial 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 also 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.

Finally, in chapter 7, 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 paper of unknown authorship. The chapter brings together many of the basics reviewed in earlier chapters, while applying various data visualization techniques using specialized Stata packages.

The final chapter concludes by briefly describing the next steps readers might take on 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://qss.princeton.press>.

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 7, “Discovery,” the contents of which include complex pieces of code and advanced techniques. 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.<sup>1</sup> 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.

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.

<sup>1</sup>Frederick Mosteller (1980). “Classroom and platform performance.” *American Statistician*, vol. 34, no. 1 (February), pp. 11–17.

Chapter title	Chapter number	Weeks
Introduction	1	1
Causality	2	2–3
Measurement	3	4–5
Prediction	4	6–7
Probability	5	8–9
Uncertainty	6	10–12
Discovery	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 foregoing outline, we can remove chapter 5, “Probability,” and chapter 6, “Uncertainty.” An alternative approach is to include the chapters on probability and uncertainty, but skip chapter 7, “Discovery,” which covers the analysis of network, spatial, and textual data.

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

### 1.3 Introduction to Stata

This section provides a brief, self-contained introduction to Stata that is a prerequisite for the remainder of this book. Stata is available from <http://www.stata.com> and has been around since 1985. The Stata Corporation prides itself on the software’s reproducibility, so code that ran in earlier versions still runs in later versions. Stata is also constantly expanding its capabilities, adding new features and commands with each version. For some changes, files created in a newer version of Stata will not necessarily work with older versions. As previously mentioned, all the commands in this book work on Stata 14. Starting with version 13, Stata can handle very long strings. Prior versions were limited to 256 characters. Stata 15 was released in June 2017 and had some new features including the ability to create markdown files and improved graphics. Stata 16 was released while this book was in its final stages. This version allows for multiple data sets in memory, together with other new features.

Stata offers several options for purchase, including plans for business, government/non-profits, education, and students. There are also three different flavors of Stata available, depending on the size of data one typically analyzes. Stata/IC can hold a maximum number of 2,048 variables and 2 billion observations. Stata/SE can handle up to 32,767 variables and include 10,998 variables in a regression. Finally, Stata/MP is the only version that takes advantage of multiprocessing. It can handle 120,000 variables and up to 20 billion observations. All three versions support the code used in this book.

Stata has an active community willing to answer questions about the software (<https://www.statalist.org/>). *The Stata Journal* (<https://www.stata-journal.com/>) (<https://www.stata.com/journals/staj.html>) likewise contains useful tips and advice, including articles that introduce new user-written

Figure 1.1. Screenshot of Stata (version 14).

commands and apply advanced analytic techniques. One of the main benefits of user-written commands from *The Stata Journal* is that they are peer reviewed.

Through numerous data analysis exercises, this book will teach you the basics of statistical programming, which will then 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. Also, we will only barely scratch the surface of what Stata is capable of, both in terms of the types of analysis possible and programming capacity. The goal is to equip readers with a foundational fluency in Stata programming that will enable them to progress to more advanced applications. Users will learn that there are often multiple ways to code the same solution to any one problem. Users interested in learning more about Stata's capabilities are encouraged to visit <https://www.stata.com/bookstore/books-on-stata/>.

In the remainder of this section, we cover four key topics: (1) arithmetic operations, (2) creating and modifying variables, (3) describing data, and (4) working with data files in Stata. Before we begin, it is necessary to first introduce Stata's user interface, captured in figure 1.1. The bottom-center Command window shows the console where we will directly type our commands. The left History window displays the commands that have been run in the current Stata session. The upper-center Results window shows the output of the executed commands. The upper-right Variables window lists all of the variables in the data set. Finally, the lower-right Properties window contains further information about the variables and dimensions of the data set, including the number of variables (columns) and number of observations (rows).

### 1.3.1 ARITHMETIC OPERATIONS

At its most basic level, Stata can be used as a calculator. We can enter any standard arithmetic operations in the Command window. For example, type `display 5 + 3` into the command line, then hit Enter on the keyboard. We insert the `display` command before the

operation so Stata shows the result directly in the Results window. Simply entering `5 + 3` will generate an error message because a valid Stata command must always come before any additional operation or code. A list of commands used in this book is included in the appendix. Each command has an abbreviated version, which we also include in the appendix for the commands we review. But we use the full command names throughout this book for clarity.

```
. display 5 + 3
8
```

Stata ignores the spaces between arithmetic operators, so `display 5+3` will return the same result as `display 5 + 3`. However, we added a space before and after the operator because it is good practice to make your code as easy to read as possible. This not only helps others to follow your code but it can also help prevent errors. As this example illustrates, a period (.) in the Results window indicates the start of the Stata command, followed immediately by the output on the next line. Importantly, this period is not part of our syntax.

Let's try some other examples. It is important for readers to try these examples on their own. Remember that we can learn programming only by doing!

```
. display 5 - 3
2
. display 5 / 3
1.6666667
. display 5 ^ 3
125
. display 5 * (10 - 3)
35
. display sqrt(4)
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 we will see in later chapters, Stata has numerous other functions. Users can obtain more information on Stata commands and functions, including available subcommands and options, by typing `help` followed by the command or function of interest into the command line. They can also use the `Help` drop-down menu or search bar in the top right of the main interface.

### 1.3.2 VARIABLES

Information is most commonly stored in *variables*. A variable is an object that takes different values across different observations. Variables will correspond to factors deemed important to our main research questions. For example, if we are interested in studying the effects of after-school programs on academic achievement, we may have variables for test scores, type of after-school program, frequency of attendance, student age, school

**Table 1.1.** 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.

demographics, and so on. To demonstrate the fundamental role variables play in any subsequent quantitative analysis, we will create a sample data set using the `input` command. Typically, we would load a data set into Stata rather than directly typing in the values of variables, but we use this simple exercise as a demonstration.

Table 1.1 contains estimates of the world population (in thousands) over the past several decades. We can enter these data directly into Stata, creating one variable for year (`year`) and another for population (`worldpop`). To ensure we are starting with an empty data set, we use the `clear` command to remove all data currently stored in memory. The `input` command must be immediately followed by a variable list when no data are in memory. This list will establish the columns of the data set. Values for variables are then entered in the same sequence in which their corresponding variable is listed. For each observation (or row), each variable value is separated by a space before entering another variable's value. Rows must be entered individually in the command line. Stata will automatically number each row, so it is only necessary to enter the values (e.g., simply type `1950 2525779` for our first row). Finally, we use `end` to tell Stata our data entry is complete; otherwise it will continue to read any input in the command line as data.

```
. clear
. input year worldpop
      year    worldpop
1. 1950  2525779
2. 1960  3026003
3. 1970  3691173
4. 1980  4449049
5. 1990  5320817
6. 2000  6127700
7. 2010  6916183
8. end
```

The `year` and `worldpop` variables are now listed in the upper-right Variables window. We can use the `list` command to show all observations with their corresponding values for year and population. The `list` command is useful here to demonstrate the structure of the data, but it becomes inefficient once we have more observations. It is important to emphasize from the start that Stata is case sensitive. Stata will consider `year` and `Year` as two separate variables.

```
. list
```

	year	worldpop
1.	1950	2525779
2.	1960	3026003
3.	1970	3691173
4.	1980	4449049
5.	1990	5320817
6.	2000	6127700
7.	2010	6916183

Now that we have a sample data set, we can examine the variable characteristics. Variables can be either numeric or string. Numeric variables contain numbers, which can include integers, decimal points, negative numbers, and so on. Numeric variables have five different storage types in Stata: `byte`, `int`, `long`, `float`, and `double`. Knowing the storage type can be important when it comes to merging data, compressing memory, or retaining all digits in large numbers. String variables, as the name implies, are composed of a string of characters. Sometimes, Stata will read numbers as strings, which could generate errors when calculations and commands require numbers. We can use the `destring` command with the `replace` option to convert a string of numbers into numeric format (e.g., `destring stringvar, replace`). This type of command is often important when we are “cleaning the data,” which is the preparation of data before it can be analyzed. Storage type and display format can be identified by using the `describe` command.

```
. describe
```

Contains data

obs:	7
vars:	2

variable name	storage type	display format	value label	variable label
year	float	%9.0g		
worldpop	float	%9.0g		

---

Sorted by:

Note: Dataset has changed since last saved.

---

We can also use the `codebook` command to obtain more detailed summary information that will prove useful in later sections, such as the range of values, number of observations that are missing data on this variable, and the frequency of example values.

```
. codebook
```

---

year

---

type: numeric (float)

range: [1950, 2010] units: 10

unique values: 7 missing .: 0/7

tabulation:	Freq.	Value
	1	1950
	1	1960
	1	1970
	1	1980
	1	1990
	1	2000
	1	2010

---

worldpop

---

type: numeric (float)

range: [2525779, 6916183] units: 1

unique values: 7 missing .: 0/7

tabulation:	Freq.	Value
	1	2525779
	1	3026003
	1	3691173
	1	4449049
	1	5320817
	1	6127700
	1	6916183

---

The results from running `codebook` tell us that, of the seven observations in our sample data set, none are missing values for `year`. In addition, `year` is stored as a numeric (float) variable, its values range from 1950 to 2010 with seven total unique values, and there is one observation for 1950, 1960, 1970, and so on. The `codebook` and `describe` commands

both show stored variable and value labels, when available. Labels are discussed in the next section.

There are also different kinds of variables. Continuous variables, such as gross domestic product (GDP), can accommodate many values. In contrast, categorical variables have a limited number of values, where each value is distinctly linked to a particular category. For example, gender, race, and political party affiliation are all categorical variables. A *binary variable*, also called a *dichotomous* or *dummy variable*, is a type of categorical variable assigned only the values of 0 (false) or 1 (true). Ordinal variables also have a limited number of values, but the values denote some type of order. Level of education may be an ordinal variable, where high school graduates could be assigned a value of 1, college graduates a value of 2, and postgraduates a value of 3; here, higher values reflect higher levels of education.

So far, we created new variables by directly inputting them. However, we can also create new variables from existing variables. To create variables in Stata, we use the `generate` command, followed by the new variable name and assigning it a value using the equals sign (=). Variable names can be up to 32 characters in length. And while they can contain numeric characters, they must start with a letter or underscore (\_). We create a new variable called `pm`, where we transform the population estimate into millions instead of thousands by dividing the `worldpop` variable by 1000. We can then see the resulting values using `list`.

```
. generate pm = worldpop / 1000
. list pm
```

	pm
1.	2525.779
2.	3026.003
3.	3691.173
4.	4449.049
5.	5320.817
6.	6127.7
7.	6916.183

Arithmetic operations can be applied only to numeric variables, but multiple variables can be included in those operations. We can add, subtract, multiply, or divide two or more variables. Here, we calculate the percentage increase in population for each decade, defined as the increase over the decade divided by its beginning population. Suppose that the population was 100,000 in one year and increased to 120,000 in the following year. In this case, we say, “the population increased by 20%.”

For this calculation, it is useful to define lagged and forward values. *Lagged values* are any values that occur in a previous time period, while *forward values* (or *leads*) are those that occur in the future. To access these lagged or forward values in Stata, we can use its

built-in indexing variable `_n`, which sequentially numbers each row or observation (e.g., the fifth row is numbered 5). To access the lagged value in the previous row, which is the prior decade in this instance, we simply subtract 1 from `_n`. To access data in proceeding rows, we add the number of desired leads to `_n` (e.g., `_n + 1` refers to the value of the following decade).

When working with lags or leads, it is important that the data are in the correct order. Consequently, it is prudent to always sort the data by the time variable using the `sort` command before processing any code involving leads or lags. Running `sort variable` places observations in ascending order according to the specified variable. To compute the percentage increase for each decade, we use `generate` to create a new variable called `lagworldpop` that stores the lagged value of `worldpop` (i.e., the population value in the preceding decade). We subtract `worldpop` by its lagged value, divide that difference by the lagged value, and multiply the result by 100 to obtain the percentage increase.

```
. sort year
. generate lagworldpop = worldpop[_n - 1]
(1 missing value generated)
. generate pctincrease = ((worldpop - lagworldpop) / lagworldpop) * 100
(1 missing value generated)
. list pctincrease
```

	pctinc~e
1.	.
2.	19.80474
3.	21.9818
4.	20.53212
5.	19.59448
6.	15.16465
7.	12.86752

We can assign variables new names using the `rename` command. This command is useful for shortening long variable names and giving variables meaningful names. Names from downloaded data sets are not always intuitive or clear (e.g., `v4` may store race). We rename our measure of population in millions to the more informative name of `popmillion`.

```
. rename pm popmillion
```

In some instances, we may want to change or reassign the values, not the names, of an existing variable. Using the `recode` command, we can directly overwrite those values, or we can store the modified values as a new variable by immediately following the command with the

`generate()` option. We previously used `generate()` as a command, but using it here as an option requires us to place the name of the new variable within the parentheses. To illustrate, we create a binary variable called `coldwar`, where 1 is assigned to years during the Cold War and 0 to the years after the Cold War.

```
. recode year (1950 1960 1970 1980 = 1) (1990 2000 2010 = 0), generate(coldwar)
(7 differences between year and coldwar)
```

We can also use the `inlist()` function to recode variables. The first argument (or value passed in the command within the parentheses) specifies the variable to be recoded. The values that follow indicate which values should be coded as 1 in a separate, newly created variable named `coldwar2`. The years excluded from the list will be recoded as 0.

```
. generate coldwar2 = inlist(year, 1950 1960 1970 1980)
```

Stata limits the total number of arguments that can be included in the `inlist()` function to 10 strings or 250 real numbers. In some cases, we will want to recode a variable to something other than 0 or 1, and this will require the `if` qualifier, which we introduce in chapter 2, along with conditional statements and the `cond()` function to demonstrate alternative ways of creating and recoding variables.

### 1.3.3 LABELS

To make our data set easier to understand, we can give variables meaningful names, but we can also assign descriptive labels to variables and to the values of categorical or ordinal variables.<sup>2</sup> *Variable labels* give a brief description of the contents of a variable. Variable labels help us keep track of what each variable means and are especially useful as the number of variables in a data set increases. As an example, we label the `pctincrease` variable with the description “Percentage increase for each decade.” Variable labels can have up to 80 characters, after which they are truncated.

```
. label variable worldpop "World population (thousands)"
. label variable popmillion "Population in millions"
. label variable pctincrease "Percentage increase for each decade"
```

*Value labels* give descriptions to individual values of a variable. Value labels allow us to more readily interpret results and keep track of what a variable’s values represent. For example, our `coldwar` variable has a value of 0 or 1. However, it is more useful to know what these values represent. A value of 0 was assigned to years following the end of the Cold War, while Cold War years have a value of 1. Consequently, we can create a set of labels using `label define`. We give this set of labels a name, followed by the label for each value. We create

<sup>2</sup>There is no need to add value labels to string variables or continuous variables, which will generally have too many values to label.

a set of labels called `cwlabel`, where 0 has the value label “After Cold War” and 1 “During Cold War.”

```
. label define cwlabel 0 "After Cold War" 1 "During Cold War"
```

Once our set of labels is defined, we attach it to a variable using `label values`. In the `label values` command, we list our variable and then the label name we created. Value labels do not replace variable values; they only attach a description to the values. The `codebook` command shows us that `coldwar` remains a numeric variable, where the numeric values are still distinct from the newly assigned labels.

```
. label values coldwar cwlabel
. codebook coldwar
```

---

coldwar		RECODE of year
type:	numeric (float)	
label:	cwlabel	
range:	[0,1]	units: 1
unique values:	2	missing .: 0/7
tabulation:	Freq.	Numeric Label
	3	0 After Cold War
	4	1 During Cold War

Because value labels are tied to specific numeric values and not to specific variables, we can use the same set of value labels on different variables. If we had a survey where many of the questions had “yes” or “no” responses, represented by values of 1 and 0, respectively, we could define a set of labels called `yesno` to apply to all of these questions. In some cases, we may need to change the label values for an already defined set of labels, such as `cwlabel`. If we assigned this label to multiple variables, we would have to tediously reassign new labels to each. Alternatively, we can simply add the `modify` option to the `label define` command to automatically update the values of an existing set of labels. In our example, we modify the respective labels for 0 and 1 to “Post-Cold War” and “Cold War.” Notice that we did not have to use the `label values` command to attach the updated value labels.

```
. label define cwlabel 0 "Post-Cold War" 1 "Cold War", modify
. list coldwar
```

	coldwar
1.	Cold War

2.	Cold War
3.	Cold War
4.	Cold War
5.	Post-Cold War
6.	Post-Cold War
7.	Post-Cold War

Finally, on occasion, we may forget the name of a variable, especially ones created by someone else. Stata offers the `lookfor` command that allows you to enter either an entire or partial string and it will return all variables that match in name or in label. This can be particularly useful in helping you identify the appropriate variables if there are several that are similar in name.

. lookfor pop	storage	display	value	
variable name	type	format	label	variable label
worldpop	float	%9.0g		World population (thousands)
popmillion	float	%9.0g		Population in millions
lagworldpop	float	%9.0g		

### 1.3.4 DESCRIBING THE DATA

For empirical analysis, we often need to know some basic information about our variables. We have already reviewed the `codebook` command, which tells us the variable's type as well as its range, number of unique values, label name, and example values when there are a larger number of observations. The `summarize` command is also very useful for providing a quick overview of the variables, displaying the variable mean, minimum and maximum values, and the number of observations with nonmissing data. It also shows the standard deviation, which we discuss more fully in chapter 2.

. summarize year worldpop pctincrease					
Variable	Obs	Mean	Std. Dev.	Min	Max
year	7	1980	21.60247	1950	2010
worldpop	7	4579529	1625004	2525779	6916183
pctincrease	6	18.32422	3.516155	12.86752	21.9818

For categorical and ordinal variables, `tabulate` is another frequently used command that displays the frequency distribution of the various categories. We could use `tabulate` on

string and continuous variables. However, when there are a large number of unique values, `tabulate` becomes inefficient.

RECODE of year			
	Freq.	Percent	Cum.
Post-Cold War	3	42.86	42.86
Cold War	4	57.14	100.00
Total	7	100.00	

We can also create a *contingency table* (also called a *cross-tabulation*) between two variables, which will show us their joint distribution.

RECODE of year			
year	Post-Cold	Cold War	Total
1950	0	1	1
1960	0	1	1
1970	0	1	1
1980	0	1	1
1990	1	0	1
2000	1	0	1
2010	1	0	1
Total	3	4	7

Further, we can combine the `tabulate` command with the `summarize()` option to calculate summary statistics by category. This is especially appropriate if you have one categorical and one continuous variable.

RECODE of year		Summary of Population in millions		
		Mean	Std. Dev.	Freq.
Post-Cold		6121.5667	797.70079	3
Cold War		3423.001	834.12731	4
Total		4579.5292	1625.004	7

### 1.3.5 DATA FILES

Before interacting with data files, we must ensure that they reside in the *working directory*. The working directory is the file path (folder) that contains the files we want to access. By default, Stata will load data from and save data to this directory. There are different ways to change the working directory. In Stata, the default working directory is shown along the bottom of the screen (see figure 1.1). Often, however, the default directory is not the directory we want to use. To change the working directory, we can use Stata's drop-down menu **File > Change Working Directory** and pick the folder from which we want to work. The `ls` command will display that directory's files and folders.

It is also possible to change the working directory using the `cd` command by specifying the full path to the folder of our choosing as a character string. Entering the `cd` command on its own, without an input, will display the current working directory. The syntax `cd "/Desktop/qss/introduction"` would set the working directory to the `introduction` subfolder. If we previously set our working directory to `cd "/Desktop/qss"`, we could simply use `cd introduction` to navigate to the `introduction` subfolder. So far, the only data we have used has been manually entered into Stata. But most of the time, we will load data from an external file. `.dta` files are data sets in Stata format. These can contain both variable labels that describe the variable and value labels that tell us how values of a variable are coded. `.dta` files might also have notes attached to them with information about the data set or the variables, such as data sources.

Once we have set our working directory, it is much easier to call upon files and avoid unnecessary retyping. Suppose that the United Nations population data in table 1.1 are saved as Stata file `UNpop.dta`, which resembles the information below.

```
year worldpop
1950 2525779
1960 3026003
1970 3691173
1980 4449049
1990 5320817
2000 6127700
2010 6916183
```

We open the file with the `use` command, which will load the data into our Stata session. Because we have already specified the working directory and the file is already in Stata format, we can simply call on the file by name. It is not necessary to type the entire path. When opening a data file with `use`, it is advisable to add the `clear` option. If there have been changes to the data set in memory, Stata will not load a new file unless those data are saved or the `clear` option is specified. We can also use the drop-down menu **File > Open > \*.dta**, which will present us with a prompt to save if there are currently unsaved data in memory.

```
. use UNpop, clear
```

A data set is a collection of variables, but we can think of it like a spreadsheet. It is often useful to visually inspect the data and its structure. We can view a spreadsheet-like representation of the data set in Stata by clicking on the Data Editor (Edit) or Data Editor (Browse) button underneath the drop-down menus (see figure 1.1). This will open a new window displaying the data. Alternatively, we can use the `browse` or `edit` commands.

Another common file type that we may encounter are CSV (comma-separated values) files. These represent tabular data and are conceptually similar to a spreadsheet of data values like those generated by Microsoft Excel or Google Sheets. 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. We can read a CSV file into Stata by going to the main drop-down menu (see figure 1.1) and clicking `File > Import > Text data (delimited, *.csv, ...)`.

We could also use the `import delimited` command. The syntax that follows loads the data into Stata's memory. Again, if we do not set the working directory, we have to specify the entire file path in quotations. Options for tailoring imported data can be found by consulting `help import delimited`. Note that when the file is not in Stata format, it is necessary to include the extension (e.g., `.csv`).

```
. import delimited using UNpop.csv, clear  
(2 vars, 7 obs)
```

Additional software options, such as `StatTransfer`, and packages within Stata itself, including `usespss`, enable users to convert a variety of file formats into Stata-readable `.dta` files. Users may also find it easier to save data from other statistical software programs as CSV or ASCII files that they can then import into Stata.

Saving data in Stata is straightforward. We just enter the command `save filename`. Should the file name already exist, we have to include the `replace` option to save over the existing file (`save filename, replace`). We can also use the drop-down menu `File` and select either `Save` or `Save As`. To delete a data set, we use `erase filename`. Deleting or saving over original data sets is not advised. Instead, when we make any changes to a data set, it should be saved as a new file, leaving our original (raw) data file intact.

### 1.3.6 MERGING DATA SETS IN STATA

Unlike in some programming packages, it is not possible to simultaneously load multiple data sets in versions of Stata that predate its most recent release (Stata 16). And sometimes, the data you need will come from various files. Consequently, two key commands you will use to combine data sets are `merge` and `append`. When we want to add additional columns (or variables) to our data set, we use `merge`. For example, we may have data on the same people or the same states in two different data sets, but different variables are contained in these files. There must be an overlap on a shared key variable or identifier, such as id number or name, to make a meaningful merger. These shared variables can be numeric or string, but values for strings must share the same exact characters and case to be matched. Although not reviewed in this book, Stata can also conduct more complex mergers based on fuzzy matches of string values.

The command structure for merge is `merge n:n varlist using filename`, where *varlist* includes the shared variables or identifier on which the merger will be based. The *n:n* argument takes on the value of either *1* or *m* (many) depending on whether the listed variables uniquely identify observations in the master data (represented by the character before the colon) or the using data (represented by the character after the colon). The *1:1* specification clarifies that we expect each observation in the master data set (the file that is open when we execute the command) to have exactly one match in the using file (specified by *filename* in the general structure). Stata accommodates matches that are not one-to-one by specifying *m*. An example is if we have a data set that contains both states and congressional districts. Our second data set may have state-level factors, including economic and demographic measures. Because there are multiple observations of each state (one observation for each congressional district) in the master data set, we would conduct a many-to-one match (*m:1*), which would merge in all of the state-level variables (creating new columns) for each row of that state.<sup>3</sup>

Let's take the `UNpop.dta` data set, which should currently be in memory, and merge in subpopulation figures from the data set `UNsubpop.dta`. This data set breaks down the world population by region. The shared identifier variable is *year*. By default, the type of match is stored in the variable `_merge`. With a normal merge, the `_merge` variable will have three values:

- 1: The observation was in the master data but not the using data.
- 2: The observation was in the using data but not the master data.
- 3: The observation was in both data sets.

We can use this information to evaluate the success of the merge. If we expect all observations to merge but there are some that have a value of 1 or 2 on the `_merge` variable, we can investigate those observations to figure out why they did not merge.

<code>. merge 1:1 year using UNsubpop</code>	
Result	# of obs.
not matched	3
from master	0 (_merge==1)
from using	3 (_merge==2)
matched	7 (_merge==3)

The output tells us that each year in the master data set was successfully matched to an observation in the using (subpopulation) data set. We also see that there were three unmatched observations in the using data set. This is because there are three years (1985, 1995, and 2005) that exist in the subpopulation data set with no match in the master data set. Stata will nonetheless merge in these observations, creating additional rows. However, these observations will have missing values for the variable `worldpop`.

<sup>3</sup>While a *m:m* is possible, even the Stata documentation does not recommend it, so in most cases the merge will be *1:1*, *1:m*, or *m:1*.

It is important to note that Stata will not change the master data unless otherwise instructed and will never replace existing values in the master data with missing data from the using data. This means that if we have the same variable in both the master and using data, Stata will ignore values of this variable in the using data, even if there are missing values in the master data. We can override this behavior with the `update` option, which replaces missing values in the master data with nonmissing values in the using data. If we additionally specify the `replace` option, Stata will replace values in the master data with nonmissing values in the using data. It will not, however, replace nonmissing values in the master data with missing values from the using data. If either `update` or `replace` are specified, the `_merge` variable can take on two additional values:

- 4: The observation was in both data sets and missing values in the master data were updated with the using data.
- 5: The observation was in both data sets and nonmissing values in the master data were replaced by nonmissing values in the using data.

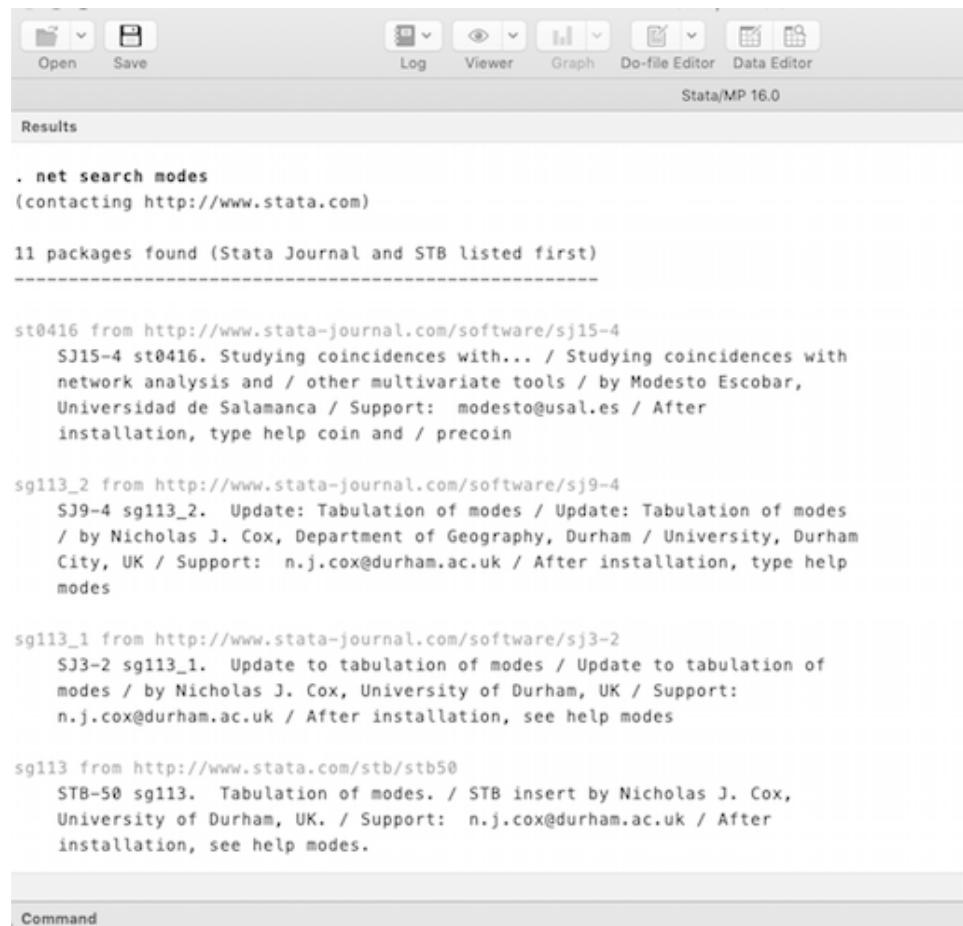
When we want to add rows to a data set where there is no overlap in values on the key identifier variable, we use the `append` command. For example, we may have yearly economic data stored in separate files by country. Because countries, the key identifier, will vary across files, we must append because `merge` is not possible. When one file is open, others can be appended with the command structure `append [varlist] using filename`. The command will add observations from a using data set to the end of the master data set. Including a variable list will only add the specified variables, but users can append an entire using file by omitting a variable list from the command. After running the `append` command, the number of observations will be equal to the number from the master data plus the number from the using data.<sup>4</sup>

Two things are important to remember about both the `merge` and `append` commands. First, prior to using the commands, one data set must already be open in Stata and it will be considered the master data set. Second, the key identifier variable must be sorted. This means that values of the variables on which the combining of files is based must be in sequential (numeric variables) or alphabetical (string variables) order. As previously introduced, we use the `sort` command to put our observations in ascending order. We can also use the `gsort` command if we need to place our observations in descending order, placing a minus sign before the sort variable (i.e., `gsort -varlist`). Observations in the current data set are ordered by year. Newer versions of Stata will automatically sort the master and using data.

### 1.3.7 PACKAGES

One of Stata's strengths is the existence of a large community of users who contribute various functionalities as Stata *packages*. Packages provide additional commands that are not included in the base version of Stata. Some of these packages first appeared in *The Stata Journal* (<http://www.stata-journal.com>) or are available from the Statistical Software Components (SSC) archive at Boston College. Throughout this book, we will employ various

<sup>4</sup>Though not used in this book, Stata offers a third way to combine two data sets using the `joinby` command, which forms all pairwise combinations within groups defined by the variables specified in the command.



The screenshot shows the Stata MP 16.0 interface with the 'Results' window active. The title bar indicates 'Stata/MP 16.0'. The menu bar includes 'File', 'Edit', 'View', 'Log', 'Viewer', 'Graph', 'Do-file Editor', and 'Data Editor'. The 'Results' window displays the output of the command '. net search modes'. The output shows 11 packages found, starting with 'st0416' and including updates like 'sg113\_2' and 'sg113\_1', and a final entry for 'sg113'. Each package entry provides a brief description and contact information.

```
. net search modes
(contacting http://www.stata.com)

11 packages found (Stata Journal and STB listed first)
-----

st0416 from http://www.stata-journal.com/software/sj15-4
SJ15-4 st0416. Studying coincidences with... / Studying coincidences with
network analysis and / other multivariate tools / by Modesto Escobar,
Universidad de Salamanca / Support: modesto@usal.es / After
installation, type help coin and / precoin

sg113_2 from http://www.stata-journal.com/software/sj9-4
SJ9-4 sg113_2. Update: Tabulation of modes / Update: Tabulation of modes
/ by Nicholas J. Cox, Department of Geography, Durham / University, Durham
City, UK / Support: n.j.cox@durham.ac.uk / After installation, type help
modes

sg113_1 from http://www.stata-journal.com/software/sj3-2
SJ3-2 sg113_1. Update to tabulation of modes / Update to tabulation of
modes / by Nicholas J. Cox, University of Durham, UK / Support:
n.j.cox@durham.ac.uk / After installation, see help modes

sg113 from http://www.stata.com/stb/stb50
STB-50 sg113. Tabulation of modes. / STB insert by Nicholas J. Cox,
University of Durham, UK. / Support: n.j.cox@durham.ac.uk / After
installation, see help modes.

Command
```

Figure 1.2. Screenshot of the Net Search Package Results. The results show the original package as well as updates.

packages. Packages from the SSC archive can be installed directly into Stata by typing `ssc install packagename`. For packages not available in the SSC archive, we use the `net search` command.

The **modes** package provides a simple illustration. Typing `net search modes` in the command line will bring up a number of options in the Results window, shown in figure 1.2. We can identify the original version of the **modes** package as *sg113* and subsequent updates are numbered with an underscore (e.g., *sg113\_2*). Clicking on the link will give us the option to install the package. We can similarly download packages using the `findit` command, which will open and display the results in the Viewer window. However, `findit` is far more thorough, as it conducts both a net search and a search through Stata's documentation.

Once the package is installed, its capabilities will be available immediately and upon opening each new instance of Stata. There is no need to call each user-written package as a library. We can obtain details on installed packages and their customized commands using `help packagename`. Typing `ado dir` will list all of the packages currently installed. And we can update all of our user-written packages by typing `adoupdate`.

Figure 1.3. Screenshot of the Stata Do-file Editor. Once we open a Stata do-file, the Do-file Editor will appear in a separate window. It can then be used to write our code.

### 1.3.8 PROGRAMMING AND LEARNING TIPS

We conclude this brief introduction to Stata by providing several practical tips for learning how to program in Stata's language. First, we should use a text editor like the one that comes with Stata to write our code rather than directly typing it into the command line. 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 command line. However, for more involved programming, it is always better to use the text editor and save our code as a text file with the .do file extension. This way, we can keep a record of our code and run it again whenever necessary.

To create a do-file, use the drop-down menu `Window > Do-file Editor > New Do-file Editor` or click the `New Do-file Editor` icon (a white square with lines and a pencil). Either approach will open a blank document for text editing in a new window where we can start writing our code (see figure 1.3). To run our code from the Do-file Editor, press the `Execute (do)` icon in the top-right corner. To run only parts of the code, highlight those lines and then press the `Execute (do)` icon. In Windows, `Ctrl+d` is an alternative shortcut for executing the code. The equivalent shortcut for Mac OS is `Command+Shift+d`. Finally, we can also run the entire code in the background (i.e., the code will not appear in the Results console) by using the drop-down menu `Tools > Execute quietly (Run)`, or by typing `Ctrl+r` in Windows or `Command+Shift+r` for Mac. Results for individual lines of code can likewise be suppressed by adding `quietly` to the beginning of the command. To execute a do-file using the command line, we use the `do` command followed by the file name.

```
. * run example do-file  
. quietly do example.do
```

As the length of our code increases, it is advised to place the command `set more off` at the top of your do-file. This will allow the code to run without interruption, assuming there are no other errors. If this command is omitted, Stata will only run some of the code and will prompt you to click “More” in the Results window before it will continue onto the next chunk of code. Depending on the size of the data set and version of Stata in use, it may also be necessary to increase the memory size (`set memory`) or the maximum number of variables (`set maxvar`).

It is sometimes useful to record a log of all activity. Typing `log using UNpop` will save all commands and output that appear in the Results window as an SMCL Stata text file. You can specify the path if you would like to save the log somewhere other than in your working directory. To close the log, simply type `log close`. Alternatively, you can open and close a log by using the `File > Log` drop-down menu, which also offers the option to view logs.

To make our code more accessible to others and readable to ourselves, we can annotate it. Adding clear comments within do-files is a good habit to establish early on, especially as our code gets more complicated. This becomes invaluable when you may later revisit the code or share your do-file with others. Stata offers several ways to write comments. First, we can place the comment within `/*` and `*/` characters. This combination is typically used to block out multiple lines of text or to deactivate larger chunks of code. Stata will consider all text following the initial `/*` as a comment, so it is necessary to add the ending `*/` when the comment ends to indicate that any further code should continue to run. We can use a simple asterisk `*` to leave a short comment (perhaps a short description of the code that will follow) or to deactivate a line of code. Similarly, comments can also follow double slashes `//` and these are particularly useful when added to the end of single command lines. Triple slashes `///` can be used to split a command line over multiple lines. Note that it is necessary to add a space between the command line and the comment indicators `//` and `///`. It is good practice to indent any lines that continue from an original command line, as illustrated in figure 1.3.

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 also essential. In the preceding examples, we place spaces around all binary operators such as `=`, `+`, and `-`, and always add a space after a comma.

Finally, starting with version 15, Stata allows Markdown, which enables us to easily embed Stata code and its output within a document using straightforward syntax in a plain-text format. The resulting documents can be produced in HTML, PDF, or even Microsoft Word format. Because Stata Markdown embeds Stata code as well as its output, the results of data analysis presented in documents are reproducible. For more information, see <https://www.stata.com/new-in-stata/markdown/>.

## 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 Stata, which is a powerful programming tool for data analysis. The remaining pages of this chapter are dedicated to exercises, designed to ensure that you have mastered the chapter's contents.

## 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 1.2 displays the names and descriptions of variables in the `turnout.dta` data file.

1. Load the data into Stata and check the dimensions of the data. How many observations are there? How many variables? Also, obtain a summary of the data. What is the range of years covered in this data set?
2. Calculate the turnout rate for each year 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?
5. Divide the data into half by election years such that you subset the data into two periods and label the value of the two periods as Early and Later years. Summarize the difference between the VEP turnout rate and the ANES turnout rate for 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

**Table 1.2.** 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)

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 Stata data files: `kenya.dta`, `sweden.dta`, and `world.dta`. The files contain population data for Kenya, Sweden, and the world, respectively. Table 1.3 presents the names and descriptions of the variables in each data set. The data are collected for a period of five 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 five-year period contributes five person-years, whereas someone who lives only through the first half of the period contributes 2.5 person-years.

- With the exception of version 16, Stata stores only one data set in memory at a time. To avoid having to open each country file individually for each calculation, append the three data files. The variable `country` distinguishes the observations from each data set. Inspect the appended data set by using the `browse` command or by clicking on the `Data Editor (Browse)` button. This allows you to view the data in a spreadsheet-like form.
- Now compute the *crude birth rate* (CBR) for a given period. The CBR is defined as

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

Calculate the CBR for each period, separately for Kenya, Sweden, and the world. Here, you can use the `bysort` command (see also section 3.6.1) to make separate

**Table 1.3.** 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)
pymen	person-years for men (in thousands)
pywomen	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.

calculations for regions and periods. For example, any command preceded by `bysort country:` will perform the calculation separately for each country. Using `bysort country period:` will provide the calculation separately for each period in each country.

Start by adding the person-years for men and women to create a variable for total person-years. Then, sum the total number of person-years and births across all age groups within each country and period using `bysort country period:` and creating two additional variables (`py_ctny_yr` and `birth_ctny_yr`, respectively). Calculate the CBR and assign appropriate labels to all variables. Finally, compute summary statistics within each country and period using cross-tabulation and briefly describe the patterns you observe.

- 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  $\delta$  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  $\delta$  is set to 5 years.

Compute the ASFR for Sweden, Kenya, and the entire world for each of the two periods and save as variable `asfr`. Assign `asfr` a meaningful variable label. Then,

using the `bysort country:` command, summarize the ASFR over age groups and period. What does the pattern of the ASFRs say about reproduction among women in Sweden and Kenya?

4. 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:

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

To begin, multiply the ASFR by 5 because the age range is 5 years and save as a new variable called `asfr5`. Compute the TFR for Sweden and Kenya as well as the entire world for each of the two periods using `bysort country period:` with the `egen` command and `sum()` function. We review the `egen` command in section 2.2.3. It provides an extension to the `generate` command, offering functions such as `sum()`, which will calculate the total sum of any variable specified within the parentheses. Here, we want to place our new variable `asfr5` within the parentheses.

We also want to limit the calculation to only reproductive women. This requires recoding `age` into a binary variable called `reproductive`, which indicates whether the age category is of reproductive age. As in the previous question, continue to assume that the reproductive age range of women is [15, 50). We confine our calculation of TFR to only women of reproductive age by including a conditional statement using the `if` qualifier at the end of our `egen` command (i.e., `egen tfr = sum(asfr5) if reproductive == 1`). We discuss conditional statements and the `if` qualifier in the next chapter.

Label the new binary and TFR variables. 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?

5. Next, we will examine another important demographic process: death. Compute the *crude death rate* (CDR), which is a concept analogous to the CBR, separately for each region in each period. The CDR is defined as

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

Label any newly created variables. Briefly describe the patterns you observe in the resulting CDRs.

6. 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 and multiply it by 1000 to display results in the thousands. Create a command to summarize the ASDR for each age group in each period, separately for each country. Focusing on just the 2005–2010 period, briefly describe the pattern you observe.

## 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 employ relational and logical operators in Stata, subset data in different ways, and compute basic descriptive statistics.

### 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.<sup>1</sup> 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. 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

<sup>1</sup>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 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)

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 in sales, administrative support, clerical, or customer services.

Let's examine the data from this experiment in detail. We begin by setting the working directory and loading the file `resume.dta` into Stata with the `use` command. You may also use the drop-down menu `File > Open`, selecting the appropriate folder and file name. Table 2.1 presents the names and descriptions of the variables in this data set.

```
. cd causality
. use resume, clear
```

The `resume` data set 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.

By viewing the Properties window or using the `describe` command, we can see that `resume` consists of 4,870 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.

```
. describe

Contains data from resume.dta
    obs:        4,870
    vars:          4
                6 Jun 2020 11:47

      storage   display     value
variable name   type   format     label     variable label
  _____
firstname       str8    %9s
sex             str6    %9s
race            str5    %9s
call            byte   %8.0g

Sorted by:
```

As discussed in chapter 1, the data set can be displayed in a spreadsheet-like format using `browse` or `edit`. Alternatively, we can look at the first several observations of the data set using the `list` command. We use the `in` option to limit the number of listed observations to only the first six. We specify the observation with which we would like to start before the slash. We start with the first observation in our example. The slash then tells Stata to include all observations up to and including that specified after the slash (in this example, the sixth observation). Before, we specified particular variables after the `list` command, but here we use Stata's wildcard character `*`, which will include all variables in the data set.

```
. list * in 1/6
```

	firstn~e	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

The results show that the second observation contains a résumé of Kristen, identified as a white female who did not receive a callback. To save this original ordering of the data, we can assign a unique identification number to each observation using the `generate` command with the `_n` indexing variable, introduced in chapter 1. We store the sequential numbering

of rows in a newly created variable called `id`. The `sort` command then allows us to return to this ordering at any point in our analysis.

```
. generate id = _n
```

We can get a summary of the data using the `modes` package installed in the previous chapter or by using the `tabulate` command. The `modes` command will show us the most frequently occurring value of a variable. Tamika is the most commonly occurring first name. Adding the `nmodes(5)` option displays the top five most frequently occurring names. For example, there were 230 résumés with Latonya as the first name of the applicant

```
. modes firstname
```

firstname	Freq.
Tamika	256

```
. modes firstname, nmodes(5)
```

firstname	Freq.
Allison	232
Anne	242
Emily	227
Latonya	230
Tamika	256

Using `tabulate`, we can also compute summary statistics for gender, race, and callback rate. Rather than typing a `tabulate` command for each variable, `tab1` allows us to truncate three `tabulate` command lines into a single line.

```
. tab1 sex race call
```

-> tabulation of sex

sex	Freq.	Percent	Cum.
female	3,746	76.92	76.92
male	1,124	23.08	100.00
Total	4,870	100.00	

-> tabulation of race			
race	Freq.	Percent	Cum.
black	2,435	50.00	50.00
white	2,435	50.00	100.00
Total	4,870	100.00	
-> tabulation of call			
call	Freq.	Percent	Cum.
0	4,478	91.95	91.95
1	392	8.05	100.00
Total	4,870	100.00	

The resulting tabulations indicate the number of résumés for each gender and race, as well as the overall proportion of résumés that received a callback (8.1%). They also show that the data set contains the same number of black and white names, while there are many more female than male applicants.

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 create a cross-tabulation summarizing the relationship between the race of each fictitious job applicant and whether a callback was received. The resulting 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).

. tabulate race call			
race	call		Total
	0	1	
black	2,278	157	2,435
white	2,200	235	2,435
Total	4,478	392	4,870

The table shows, for example, that among 2,435 ( $= 2,278 + 157$ ) résumés with stereotypically black names, only 157 received a callback. Adding the `row` option computes the callback rate, or the proportion of those who received a callback, for the entire sample and for black and white applicants. As recommended in chapter 1, we annotate our code with comments, which are preceded by an asterisk (\*).

```
. * overall callback rate: total callbacks divided by the sample size
. tabulate race call, row
```

Key	
	frequency
	row percentage
race	call
	0 1
black	2,278 157
	93.55 6.45
white	2,200 235
	90.35 9.65
Total	4,478 392
	91.95 8.05
	100.00

From this analysis, we observe that the callback rate for the résumés with African American-sounding names is .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.

Rather than looking at the proportion of a sample falling into each cross-category (or cell), we may want to directly compute the mean of one variable across categories, particularly when we are dealing with continuous variables. For this, we combine the `tabulate` command with the `summarize()` option to view the callback rate for each race. Here, `call` is a *binary variable* that takes the value of 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, but our key outcome or variable of interest will not always be binary. Using `tabulate` with `summarize` additionally provides us with the standard deviation of the callback rate for each race, which we discuss later.

Summary of call			
race	Mean	Std. Dev.	Freq.
black	.06447639	.24565008	2,435
white	.09650924	.295349	2,435
Total	.08049281	.27208256	4,870

To calculate separate means by race, we could also use the `summarize` command together with the `if` qualifier, which we introduce in the next section.

## 2.2 Subsetting the Data in Stata

In this section, we learn how to subset a data set in various ways. We first introduce relational operators such as “greater than” (`>`), “less than” (`<`), and “equal to” (`==`). We then review logical operators, including “AND” and “OR.” We show how relational and logical operators can be used to specify the observations and variables to be extracted from a data set. They can also be used to generate new, or replace values of existing, variables. At the end of this section, we demonstrate how the `keep` and `drop` commands allow us to select or drop observations based on specified conditions.

### 2.2.1 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” (`!=` or `~`). These operators return logical values of 1 if `true` and 0 if `false`.

As reviewed in chapter 1, we use a single equals sign (`=`) to assign a variable a numeric or string value. The expression `generate x = 2`, for example, tells Stata to create a variable named `x` and assign to it the value of 2. A double equals sign (`==`), on the other hand, asks Stata to evaluate the expression that the first object is equal to the second object. If the condition is met, Stata returns a value of 1. If the condition is not met, Stata returns a value of 0. Using the `display` command, we can see that Stata returns a 0 or 1 for the relational operators we have introduced. We place parentheses around the string comparisons because Stata will otherwise read the initial double quotations as the start of a string to be displayed in the Results window, generating an error message.

```
. display 4 == 4
1

. display 4 > 3
1

. display ("Hello" == "hello") // Stata is case sensitive
0

. display ("Hello" != "hello")
1
```

We can also apply relational operators to all observations of a variable in order to quickly generate new variables by using the *if qualifier*. When applied to a variable, the operators evaluate each observation separately. If an observation satisfies the relational condition that follows the `if` qualifier, Stata performs a specified operation on the observation. If the condition is not met, Stata does nothing. We create the binary variable `abovefour` to indicate whether the person’s identification number, which we created earlier using notation `_n`, is greater than or equal to 5. Because of our specification in this instance, Stata will assign a

value of 1 only when a statement is true; it will assign a missing value if false. Listing the first 10 consecutive observations confirms our conditional assignment.

```
. generate abovefour = 1 if id >= 5
(4 missing values generated)

. list id abovefour in 1/10
```

	id	abovef~r
1.	1	.
2.	2	.
3.	3	.
4.	4	.
5.	5	1
6.	6	1
7.	7	1
8.	8	1
9.	9	1
10.	10	1

### 2.2.2 LOGICAL OPERATORS

As previously shown, Stata evaluates relational operators with logical values of 1 if true and 0 if false. We can also produce logical values with the *logical operators* & and | , respectively corresponding to *logical conjunction* (“AND”) and *logical disjunction* (“OR”). The value of “AND” (&) is TRUE only when all specified conditions have a value of TRUE. The syntax below illustrates how Stata deals with different logical constructs. Because there are multiple conditions in our statement, we place parentheses around each for increased legibility.

```
. * FALSE & TRUE
. display (5 == 4) & (5 < 10)
0

. * TRUE & TRUE
. display (5 > 4) & (5 < 10)
1
```

The value of “OR” (|) is TRUE when at least one of the conditions has the value TRUE.

```
. * FALSE | TRUE
. display (5 == 4) | (5 < 10)
1
```

**Table 2.2.** Logical Conjunction “AND” and Disjunction “OR”

Statement <i>a</i>	Statement <i>b</i>	<i>a</i> AND <i>b</i>	<i>a</i> OR <i>b</i>
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.

```
. * FALSE | FALSE
. display (5 == 4) | (5 == 10)
0
```

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
. display (5 > 1) & (5 == 6) & (7 < 10)
0
```

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

```
. * (TRUE | FALSE) & FALSE - the parentheses evaluate to TRUE
. display (5 > 4 | 5 == 3) & (7 < 5)
0

.

. * TRUE | (FALSE & FALSE) - the parentheses evaluate to FALSE
. display (5 > 4) | (5 == 3 & 7 < 5)
1
```

As we show in the next section, we can not only combine the logical operations “AND” and “OR,” but we can also include different conditions for more than one variable. We use these conditions to create new variables and to subset the data.

### 2.2.3 SIMPLE CONDITIONAL STATEMENTS AND VARIABLE CREATION

We can create new variables that store the results of relational and logical operations. For example, we can apply a simple conditional statement using the `cond()` function to create

a new binary variable called `BlackFemale`. This variable will equal 1 if the job applicant's name sounds black and female, and 0 if otherwise. The function `cond(X, Y, Z)` contains three elements. For each observation in X that is TRUE, the corresponding value listed in Y is returned. In contrast, for each observation in X that is FALSE, the corresponding value in Z is returned. We create a numeric variable `BlackFemale` with the following syntax.

```
. generate BlackFemale = cond(race == "black" & sex == "female", 1, 0)
```

The first argument within the parentheses `race == "black" & sex == "female"` sets the logical and relational conditions (i.e., black AND female-sounding names) to be tested. The next arguments respectively specify the values to be assigned to the `BlackFemale` variable should the conditions be true 1 or false 0. Stata will code any observations where the condition is not true as 0. Even observations with missing values will be coded as 0. But we might want an observation that has a missing value for `race` or `sex` to also have a missing value for `BlackFemale` since we do not know whether both conditions are met. In Stata, missing values are denoted by a period (.) for numeric variables and double quotations ("") for string variables. We can use the `replace` command to recode `BlackFemale` as missing if either variable has a missing value. We discuss missingness later in chapter 3.

```
. replace BlackFemale = . if race == "" | sex == ""
(0 real changes made)
```

Stata informed us that zero changes were made, which means none of the observations in our data set were missing values on `race` or `sex`. Using `tabulate` with the `if` qualifier, we can then see that the `BlackFemale` variable equals 1 only when a résumé belongs to an African American female, and 0 otherwise.

```
. * confirming the accuracy of our code
. tabulate race sex if BlackFemale == 0

      sex
      female     male |   Total
-----+-----+-----
    black |       0     549 |    549
   white |  1,860     575 |  2,435
-----+-----+-----
    Total |  1,860    1,124 |  2,984

. tabulate race sex if BlackFemale == 1

      sex
      female |   Total
-----+-----+
    black |  1,886    1,886
-----+-----+
    Total |  1,886    1,886
```

In addition to binary variables, we may want to create a new *factor variable* (or *factorial variable*) in Stata using relational and logical operators. 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 called `type` that takes a numeric value for each of the four combinations of race and gender (i.e., black female, black male, white female, white male). We start by creating a variable assigned all missing values. We assign a new value with each race-gender category and label the variable values as reviewed in chapter 1.

```
. generate type = .
(4,870 missing values generated)

. replace type = 1 if race == "black" & sex == "female"
(1,886 real changes made)

. replace type = 2 if race == "black" & sex == "male"
(549 real changes made)

. replace type = 3 if race == "white" & sex == "female"
(1,860 real changes made)

. replace type = 4 if race == "white" & sex == "male"
(575 real changes made)

.

. label define typecat 1 "Black Female" 2 "Black Male" 3 "White Female" 4 "White
Male"

. label values type typecat
```

The `tabulate` command can be applied to obtain the number of observations that fall into each level. We can likewise see the numeric value assigned to each level by adding the `nolabel` option.

```
. * number of observations for each level
. tabulate type

      type |      Freq.     Percent      Cum.
    _____
    Black Female |    1,886      38.73      38.73
    Black Male |      549      11.27      50.00
    White Female |    1,860      38.19      88.19
    White Male |      575      11.81      100.00
    _____
          Total |    4,870     100.00
    _____

.

. tabulate type, nolabel

      type |      Freq.     Percent      Cum.
    _____
          1 |    1,886      38.73      38.73
```

2	549	11.27	50.00
3	1,860	38.19	88.19
4	575	11.81	100.00
<hr/>			
Total	4,870	100.00	

Stata offers a shortcut `group()` function that can be used with the `egen` command to similarly store all combinations between two or more variables in a factor (or categorical) variable. The `label` option retains the value labels attached to the grouping variables. When using this option, then, it is important that the variables used for grouping have meaningful value labels.

. egen type2 = group(race sex), label			
. tabulate type2			
group(race sex)	Freq.	Percent	Cum.
<hr/>			
black female	1,886	38.73	38.73
black male	549	11.27	50.00
white female	1,860	38.19	88.19
white male	575	11.81	100.00
<hr/>			
Total	4,870	100.00	

The `egen` command provides extensions to the `generate` command by offering functions that perform calculations across observations. This command allows users to calculate sample means, counts, standard deviations, and more. A list of the many possible functions with instructions on their use can be found by typing `help egen`. The `egen` command also allows users to compute a number of functions across variables within an observation by adding `row` to the statistic, such as `rowmean` or `rowsd`.

#### 2.2.4 SUBSETTING USING CONDITIONS

In addition to storing the results of relational and logical conditions in variables, we can use these operations to subset the data. To illustrate, we may want to know the callback rate for just black-sounding names in our sample. In section 2.1, we used the `tabulate` and `summarize` commands. However, we can also use the `if` qualifier, which allows us to specify relational and logical operations in our summary statistics. For example, instead of using `tabulate race, summarize(call)`, we can use `summarize` with `if` to compute the callback rate among the résumés with black-sounding names. We can similarly use `if` with the `tabulate` command. Keep in mind that `race` is a string variable, so we need to include quotations around the variable value.

. * callback rate for black-sounding names					
. summarize call if race == "black"					
Variable	Obs	Mean	Std. Dev.	Min	Max
call	2,435	.0644764	.2456501	0	1
. tabulate call if race == "black"					
call	Freq.	Percent	Cum.		
0	2,278	93.55	93.55		
1	157	6.45	100.00		
Total	2,435	100.00			

Both commands subset the `call` variable to observations where the value for `race` is equal to `black`. They provide summary statistics based on the subset of the data, both revealing that the mean callback rate for a black-sounding name is approximately 6.4%.

We can also create multiple logical conditions across different variables to subset the data. For example, we can calculate summary statistics for observations where `race` is equal to `black` and `sex` is equal to `female` by using `if` with the “AND” logical operator. Alternatively, we can use our newly created binary variable `BlackFemale`.

. * callback rate for females with black-sounding names					
. summarize call if race == "black" & sex == "female"					
Variable	Obs	Mean	Std. Dev.	Min	Max
call	1,886	.0662778	.2488331	0	1
. summarize call if BlackFemale == 1					
Variable	Obs	Mean	Std. Dev.	Min	Max
call	1,886	.0662778	.2488331	0	1

## 2.2.5 PRESERVING AND TRANSFORMING DATA SETS

As shown, we can calculate descriptive statistics by subgroup using the `if` qualifier. However, we can also subset the data by altering the rows or columns of the data set itself, dropping observations or variables we do not need. The `keep` command can be used to limit the data set to only certain variables or, when used with the `if` or `in` options, to subset the data to only observations meeting specified conditions. Similarly, the `drop` command will remove listed variables from the data set or will drop observations that meet conditions set by the `if` or `in` statements.

To continue our example, if we want our data set to only include résumés with black-sounding names, we would use the command `keep if race == "black"` or `drop if race != "black"`. Once we drop the nonblack observations, we can directly compute summary statistics for black-sounding names using only commands like `summarize`. The `if` qualifier is no longer necessary.

We may want to drop particular variables from our data set, such as `firstname` or `abovefour` (`drop firstname abovefour`). Users must be sure not to drop variables that are needed in conditions that specify which observations to keep (or drop). For example, if we drop the `race` variable, and then try to subset the data by keeping all observations where `race` is equal to `black`, we will get an error because that variable no longer exists.

We can also create a summary data set using the `collapse` command. This command aggregates the data, providing a summary of the sample means, medians, percentiles, sums, minimum or maximum values, standard errors, and standard deviations. Users can specify which of these statistics they would like, and for which variables, by placing the Stata name for the statistic in parentheses, followed by the desired variables. For example, `collapse (mean) call (median) call (p25) call` would provide the mean, median, and 25th percentile for our `callback` variable. We could add additional variables, and the full list of statistics we could include in our command can be found by typing `help collapse`. When no statistic is specified in the command, the default is to return the mean. Users can also request that these statistics be calculated by subgroup using the `by()` option (e.g., `collapse call, by(race)`). Multiple variables can be included in the `by()` specification, such as `sex` and `race` to obtain summary statistics for every sex–race combination.

It is important to note that the `keep`, `drop`, and `collapse` commands will fundamentally transform our data set. If we wish to access the original, full data set again, we will have to reload it. It is good practice to not write or save over original data. Running these commands on their own risks overwriting the original data if we accidentally hit Save. Consequently, the `preserve` and `restore` commands are helpful if we do not want to create new data sets. The `preserve` command essentially takes a snapshot of our data when the command is issued. We can make whatever changes we want to the data and then return to the preserved version of the data by issuing the `restore` command.<sup>2</sup>

Let's use the `keep` command to calculate the callback rate for black-sounding names. We keep only the observations of résumés with black-sounding names and then compute the callback rate using `summarize`. We use the `preserve` and `restore` commands so the original data set is left intact. We first use the `summarize` command on the full sample for comparison to the full-sample mean and number of observations prior to using `keep`.

```
. * original number of observations
. summarize call
```

Variable	Obs	Mean	Std. Dev.	Min	Max
call	4,870	.0804928	.2720826	0	1

<sup>2</sup>If we are using `preserve` within a do-file, the data will be restored when the do-file stops running or when the `restore` command is issued, whichever comes first.

```
.
. * subset blacks only
. preserve
. keep if race == "black"
(2,435 observations deleted)
. summarize call

      Variable |       Obs        Mean    Std. Dev.      Min      Max
                 |   2,435  .0644764  .2456501          0          1
. tabulate race

      race |      Freq.     Percent      Cum.
                 |   2,435      100.00    100.00
                 |
      Total |      2,435      100.00

. restore
```

We can see that the number of observations dropped from 4,870 to 2,435 after using the `keep` command. The `tabulate` command confirms that only black-sounding names are kept in the data set.

Because the original data set has been restored, we can also use the `collapse` command to create a summary data set that contains the mean for all specified variables. As mentioned, `collapse` can be used to calculate other statistics, but the default calculation is the mean. Readers may find it useful to use `browse` or `edit` to examine the data set both before and after running the `collapse` command to see how the data set is transformed. Here, we demonstrate how we can calculate the mean callback rate by sex and race. We again use the `preserve` and `restore` commands to restore the original data.

```
.
. preserve
. collapse call, by(sex race)
. list

      sex      race      call
      1. female    black    .066278
      2. female   white    .098925
      3. male     black    .058288
      4. male     white    .088696

. restore
```

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.

### 2.3 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). Using `sort` to return the data set to its original order, we see that 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).

```
. * restore data to original order and view first observation
. sort id
. list firstname call in 1
```

	firstname	call
1.	Allison	0

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 in which 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 decided to raise the minimum wage and in this state the unemployment rate increased afterwards. This does not necessarily imply that a higher minimum wage led to the increase in the unemployment rate. 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

**Table 2.3.** Potential Outcome Framework of Causal Inference.

Résumé $i$	Black-sounding name sounding name $T_i$	Callback		Age	Education
		$Y_i(1)$	$Y_i(0)$		
1	1	1	?	20	College
2	0	?	0	55	High school
3	0	?	1	40	Graduate school
:	:	:	:	:	:
$n$	1	0	?	62	College

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 name,  $T_i = 0$ , 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 were used,  $Y_i(0)$ . For every résumé, only one of the two potential outcomes is observed and the other is missing (indicated by "?").

received job training secured a job afterwards, it does not necessarily follow that it was the job training program that 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  $i$ th 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 able to observe only 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. 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 that 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 6.

### 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.,  $Y_i(1) - Y_i(0)$ ):

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

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.

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 individuals 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.

#### 2.4.2 SOCIAL PRESSURE AND VOTER TURNOUT

We consider a study of peer pressure and voter turnout.<sup>3</sup> 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.

<sup>3</sup>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.

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

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!

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

**Table 2.4.** Social Pressure Experiment Data.

Variable	Description
hhszie	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)

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 the social pressure message, the civic duty message, or the Hawthorne effect message. The experiment also has a control group that consists of those voters receiving no message. The researchers randomly assigned each voter to one of the four groups and examined whether voter turnout was different across the groups.

Now that we understand the design of this experiment, let us analyze the data. The data file is named `social.dta`, which we load into Stata via the `use` command. Table 2.4 displays the names and descriptions of the variables in the social pressure experiment data. Using the `compact` option with the `codebook` command provides a brief summary of the data, including a list of variables with their number of nonmissing observations, total number of unique values, means, and minimum and maximum values.

. use social, clear						
. codebook, compact						
Variable	Obs	Unique	Mean	Min	Max	Label
sex	305866	2	.	.	.	.
yearofbirth	305866	86	1956.214	1900	1986	
primary2004	305866	2	.4013784	0	1	
messages	305866	4	.	.	.	
primary2006	305866	2	.3122446	0	1	
hhszie	305866	8	2.184421	1	8	

As before, we use `tabulate` with `summarize()` to compute turnout for each treatment group. The outcome variable of interest is `turnout` in the 2006 primary election, which is coded as the binary variable `primary2006`, where 1 represents having cast a vote in 2006 and 0 is abstention.

```
. * turnout for each group
. tabulate messages, summarize(primary2006)
```

messages	Summary of primary2006		
	Mean	Std. Dev.	Freq.
Civic Duty	.31453765	.46433755	38,218
Control	.29663831	.45677687	191,243
Hawthorne	.32237462	.46739164	38,204
Neighbors	.37794822	.48488093	38,201
Total	.31224458	.46340976	305,866

By using the `if` qualifier with `summarize`, we can obtain the mean turnout and other summary statistics for just the `control` group. Stata stores the results of these calculations and they are viewable using the `return list` command.

```
. * turnout for control group
. summarize primary2006 if messages == "Control"
```

Variable	Obs	Mean	Std. Dev.	Min	Max
primary2006	191,243	.2966383	.4567769	0	1

```
. return list

scalars:
        r(N) = 191243
        r(sum_w) = 191243
        r(mean) = .2966383083302395
        r(Var) = .2086451133559862
        r(sd) = .4567768748043033
        r(min) = 0
        r(max) = 1
        r(sum) = 56730
```

The values held in `r()` refer to return (or `r`-class) results from general commands. As we will see in later chapters, Stata also stores the (`e`-class) results of estimation commands, found in `e()`, which we can similarly view by typing `ereturn list`. By running `return list` after our `summarize` command, we can see that the control group's baseline turnout is stored under `r(mean)`. A number of other statistics are stored as well, including the number of observations, `r(N)`, and the sum of a variable, `r(sum)`. Referring to these stored results

reduces the risk of mistyping values. It is important to note, though, that every time we run a new `summarize` command, the stored values will be replaced with the most recent results.

Subtracting the baseline turnout of the control group gives the average causal effect of each message. Though we could do this manually by subtracting values in the tabulation table, we subtract the control mean stored in `r(mean)` and save the result in a new variable called `votediff`. We repeat our `tabulate` command for each of the treated groups to view the average treatment effect.

```
. * subtract control group turnout from each group
```

```
. generate votediff = primary2006 - r(mean)
```

```
. tabulate messages if message!="Control", summarize(votediff)
```

messages	Summary of votediff		
	Mean	Std. Dev.	Freq.
Civic Duty	.01789934	.46433755	38,218
Hawthorne	.02573631	.46739164	38,204
Neighbors	.08130991	.48488093	38,201
Total	.04164458	.47312739	114,623

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 (`hhsiz`e).

We have demonstrated how the `tabulate` command combined with `summarize()` provides summary statistics when dealing with one categorical variable and one key variable. However, in this instance, we want to show summary statistics across multiple variables for multiple groups. Stata provides the `tabstat` command for such scenarios. This command also offers a wider selection of calculations that go beyond means, including sum, range, standard deviation, count, median, and more (see `help tabstat`). The `table` command is similar but offers a bit more customization in terms of table display.

First we create a variable for age by subtracting the year of birth from the election year, which is 2006. We then show the means of this and other variables for groups with different messages using `tabstat`.

```
. generate age = 2006 - yearofbirth
```

```
. tabstat age primary2004 hhsiz, by(message) statistics (mean)
```

messages	age	pri~2004	hhszie
Civic Duty	49.65904	.3994453	2.189126
Control	49.81355	.4003388	2.183667
Hawthorne	49.7048	.40323	2.180138
Neighbors	49.85294	.4066647	2.18777
Total	49.78558	.4013784	2.184421

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.<sup>4</sup> 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 earlier, answering this question requires inference about the NJ employment rate in the absence of such a raise in the minimum wage. Since this 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

<sup>4</sup>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

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.

We load the Stata file `minwage.dta`. As before, the `describe` command shows the number of observations and the number of variables (i.e., dimensions of the data), and the `codebook` command with the `compact` option provides a brief statistical summary of each variable. Table 2.5 displays the names and descriptions of the variables in the minimum-wage study data.

```
. use minwage, clear
. describe
Contains data from minwage.dta
obs: 358
vars: 8                               28 Dec 2019 20:21

      storage   display    value
variable name  type    format   label   variable label
chain          str10   %10s
location        str9    %9s
wagebefore     float   %9.0g   wageBefore
wageafter      float   %9.0g   wageAfter
fullbefore     float   %9.0g   fullBefore
fullafter      float   %9.0g   fullAfter
partbefore     float   %9.0g   partBefore
partafter      float   %9.0g   partAfter
```

```
Sorted by:
```

```
. codebook, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
chain	358	4	.	.	.	
location	358	5	.	.	.	
wagebefore	358	31	4.617709	4.25	5.75	wageBefore
wageafter	358	22	4.993855	4.25	6.25	wageAfter
fullbefore	358	46	8.47486	0	60	fullBefore
fullafter	358	40	8.361732	0	40	fullAfter
partbefore	358	58	18.75419	0	60	partBefore
partafter	358	56	18.68855	0	60	partAfter

Because we want to make comparisons between PA and NJ in our analysis, we first use the `generate` command combined with the `cond()` function to create a state variable that makes a distinction between PA and NJ. This allows us to group all NJ locations together when we make our comparisons to PA. We also use this variable to subset the data with the `if` qualifier. Recall that the `cond()` function tells Stata what to assign the new variable if the specified condition (`location == "PA"`) is respectively true (assign "PA") or false (assign "NJ").

```
. * create a dichotomous state variable
. generate state = cond(location == "PA", "PA", "NJ")
```

To make sure that the restaurants followed the law, we next examine whether the minimum-wage actually increased in NJ after the law was enacted. We 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 wages before and after the NJ law went into effect. We use the `cond()` function to create two binary variables that indicate whether the minimum wage was lower than the new rate of \$5.05 both before and after the law. Because these are binary variables, with values of 0 or 1, the means can be interpreted as the proportion of restaurants whose wage is less than \$5.05 at each period. The `tabstat` command shows us these proportions broken down by state when we include the `by` option.

```
. * proportion of restaurants whose wage is less than $5.05
. generate minwagebefore = cond(wagebefore < 5.05, 1, 0)
. generate minwageafter = cond(wageafter < 5.05, 1, 0)
. tabstat minwagebefore minwageafter, by(state) statistics(mean)
```

Summary statistics: mean by categories of: state		
state	minwag_e	minwag_r
NJ	.9106529	.0034364
PA	.9402985	.9552239
Total	.9162011	.1815642

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 a minimal impact on the wages in PA restaurants. The analysis shows that the NJ restaurants followed the law by increasing their wage to meet the new minimum of \$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
. generate fullpropafter = fullafter / (fullafter + partafter)
```

To compute the difference in state means, we use the `summarize` command with the `if` qualifier. As introduced earlier, we use the results stored in `r(mean)`. Because stored results are replaced with the processing of subsequent commands, we save the `r(mean)` result into what is called a *scalar*. A scalar stores only one piece of information or value, unlike other variables in our data, which are available for each row (or observation) and form their own columns in the data set. Scalars do not become part of the data set. We create a scalar using the `scalar` command, assigning it a single value, whether a directly specified number or string, or a value taken from stored results (e.g., `r(mean)`). Saving the mean in a new variable using the `generate` command would unnecessarily create a new column in the data set. Scalars allow us to simply store the value in memory. If a variable and scalar have the same name, Stata will assume the user means the variable, so it is important to assign scalars unique names. Users can alternatively place the pseudofunction `scalar()` around the scalar name so Stata makes the correct interpretation. However, the code will look cleaner and it is better form if users avoid assigning the same names to variables and scalars.

```

. * compute the difference-in-means
. summarize fullpropaftter if state == "NJ"

      Variable |       Obs        Mean    Std. Dev.      Min      Max
fullpropaf_r |     291     .320401     .2510016          0          1
. scalar fullpropaftterNJ = r(mean)
.

. summarize fullpropaftter if state == "PA"

      Variable |       Obs        Mean    Std. Dev.      Min      Max
fullpropaf_r |      67     .2722821     .2472422          0     .9032258
. scalar fullpropaftterPA = r(mean)
.

. display fullpropaftterNJ - fullpropaftterPA
.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 aforementioned 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

businesses self-select into treatments. The highlighted 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 the 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 `tabulate` command, specifying the `column` and `nofreq` options, giving us the proportion of restaurants in each state. Omitting `nofreq` would also display the actual number of observations across states, not just the proportion. The `column` option defines proportions by column (i.e., the proportion of each chain in NJ, then the proportion of each chain in PA, with columns for each state adding to 100). Alternatively, adding the `cell` option at the end of the `tabulate` command would give the total proportion across all cells, not per column (i.e., across all states and chains, with all cells, not individual columns, adding to 100).

```
. tabulate chain state, column nofreq
```

chain	state		Total
	NJ	PA	
burgerking	40.55	46.27	41.62
kfc	22.34	14.93	20.95
roys	25.09	22.39	24.58
wendys	12.03	16.42	12.85
Total	100.00	100.00	100.00

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.

```
. * subset Burger King only, by state
. summarize fullpropaftter if state == "NJ" & chain == "burgerking"
```

Variable	Obs	Mean	Std. Dev.	Min	Max
fullpropaf~r	118	.3577372	.2608763	0	1

```
. scalar njbk = r(mean)
. summarize fullpropaftter if state == "PA" & chain == "burgerking"
```

Variable	Obs	Mean	Std. Dev.	Min	Max
fullpropaf~r	31	.3212979	.2603062	0	.8433735

```
. scalar pabk = r(mean)
.
. * Burger King only NJ vs PA comparison of full-time employee rates
. display njbk - pabk
.03643934
```

Limiting the comparison to just Burger King restaurants yields a finding similar to our prior analysis, suggesting that the fast-food chain may not be a confounding factor.

Another possible confounder is the location of restaurants. NJ Burger King restaurants closer to PA may yield a more credible comparison, perhaps because their local economies share similar characteristics. To address this possible confounding bias, we subclassify the data by restaurant location. Specifically, we focus on Burger King restaurants located in northern and southern NJ that are near PA, while excluding those in the Jersey shore and central New Jersey. Because we are running multiple `summarize` commands to calculate differences

in means, we again store each mean result `r(mean)` in a scalar before they are overwritten by the next command. This analysis adjusts for both the type of restaurants and their locations through statistical control. Like the period, the “greater than” sign that precedes our third line of code is not part of the syntax. It indicates a line of code that continues from an initial command in the Results window itself.

```
. * subset Burger King, by NJ location
. summarize fullpropaf after if state == "NJ" & chain == "burgerking" & ///
>           (location != "shoreNJ" & location != "centralNJ")

Variable |      Obs       Mean     Std. Dev.      Min      Max
fullpropaf_r |      89    .3527964    .2615981        0        1
. scalar njbk_subset = r(mean)
.
. * Burger King only and excluding NJ shore and central NJ
. display njbk_subset - pabk
.03149854
```

The results show that even within the smaller subset of New Jersey locations, 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 that 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, we use this design to compare the same set of fast-food restaurants in NJ before and after the minimum wage was raised.

```

. * full-time employment proportion in the previous period for NJ
. generate fullpropbefore = fullbefore / (fullbefore + partbefore)
. summarize fullpropbefore if state == "NJ"

Variable |      Obs       Mean    Std. Dev.      Min      Max
fullpropbe~e |     291    .2965262    .2304592          0          1
. scalar fullpropbeforeNJ = r(mean)
.
. * mean difference between before and after the minimum wage increase
. display fullpropafternJ - fullpropbeforeNJ
.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 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

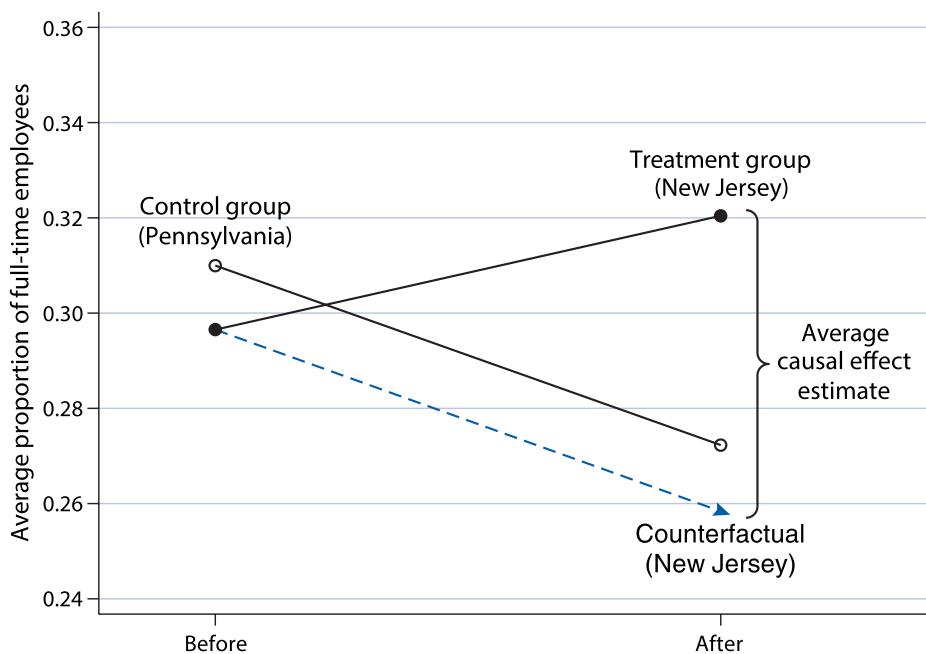


Figure 2.2. The Difference-in-Differences Design in the Minimum Wage Study. The observed outcomes, the average proportion of full-time employees, are shown before and after the minimum wage was increased for the treatment group (fast-food restaurants in New Jersey; solid black circles) and the control 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 time trend for the control group. The estimated average causal effect for New Jersey restaurants is indicated by the curly brace.

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.<sup>5</sup> 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.

<sup>5</sup>Formally, the sample average treatment effect for the treated (SATT) is the sample average of individual-level causal effect among the treated units,  $SATT = \frac{1}{n_1} \sum_{i=1}^n T_i(Y_i(1) - Y_i(0))$ , 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.

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):

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

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.

```
. * full-time employment proportion in the previous period for PA
. summarize fullpropbefore if state == "PA"

Variable |      Obs       Mean    Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----+
fullpropbe~e |      67     .3099657     .2397895      0     .9493671
.
. scalar fullpropbeforePA = r(mean)
.
. * differences in differences using mean difference between before and after, by
state
. display (fullpropafternJ - fullpropbeforeNJ) - (fullpropaftterPA -
fullpropbeforePA)
.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 variable is 4.5.

While both the mean and median measure the center of the distribution, the mean is more sensitive to *outliers*. 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. 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

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

where  $x_{(i)}$  denotes the value of the  $i$ th 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 assess 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. To calculate the median, we use `summarize` with the `detail` option.

The median will then be stored in p50, which represents the 50th percentile summary statistic.

```

. * cross-section comparison between NJ and PA median
. summarize fullpropaftter if state == "NJ", detail

fullpropaftter

```

---

Percentiles	Smallest		
1%	0	0	
5%	0	0	
10%	0	0	Obs
25%	.1111111	0	Sum of Wgt.
50%	.2916667		Mean
		.8571429	Std. Dev.
75%	.5	1	Variance
90%	.6666667	1	Skewness
95%	.7777778	1	Kurtosis
99%	1	1	

```

. scalar medaftterNJ = r(p50)

.

.

. summarize fullpropaftter if state == "PA", detail

fullpropaftter

```

---

Percentiles	Smallest		
1%	0	0	
5%	0	0	
10%	0	0	Obs
25%	.0952381	0	Sum of Wgt.
50%	.21875		Mean
		.7894737	Std. Dev.
75%	.3846154	.7924528	Variance
90%	.7142857	.8433735	Skewness
95%	.7894737	.9032258	Kurtosis
99%	.9032258	.9032258	

```

. scalar medaftterPA = r(p50)

.

.

. display medaftterNJ - medaftterPA
.07291666

.

.

. * before and after comparison
. summarize fullpropbefore if state == "NJ", detail

```

fullpropbefore				
	Percentiles	Smallest		
1%	0	0		
5%	0	0		
10%	0	0	Obs	291
25%	.1071429	0	Sum of Wgt.	291
50%	.2666667		Mean	.2965262
		Largest	Std. Dev.	.2304592
75%	.4736842	.7936508		
90%	.6363636	.8125	Variance	.0531115
95%	.7	.8571429	Skewness	.4607855
99%	.8125	1	Kurtosis	2.297936
.	scalar medbeforeNJ = r(p50)			
.				
.	display medafterNJ - medbeforeNJ			
.	.02499998			
.				
.	* median difference-in-differences			
.	summarize fullpropbefore if state == "PA", detail			
fullpropbefore				
	Percentiles	Smallest		
1%	0	0		
5%	0	0		
10%	0	0	Obs	67
25%	.125	0	Sum of Wgt.	67
50%	.2307692		Mean	.3099657
		Largest	Std. Dev.	.2397895
75%	.4827586	.75		
90%	.6923077	.7575758	Variance	.057499
95%	.75	.8095238	Skewness	.6968449
99%	.9493671	.9493671	Kurtosis	2.555775
.	scalar medbeforePA = r(p50)			
.				
.	display (medafterNJ - medbeforeNJ) - (medafterPA - medbeforePA)			
.	.03701921			

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 `summarize` command used with `detail` will display values for each quartile, along with the minimum, mean, and maximum values and other descriptive statistics. 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 using the `tabstat` command with the `statistic(iqr)` option.

```
. * summary shows quartiles as well as minimum, maximum, and mean
. summarize wagebefore if state == "NJ", detail
```

#### wageBefore

	Percentiles	Smallest		
1%	4.25	4.25		
5%	4.25	4.25		
10%	4.25	4.25	Obs	291
25%	4.25	4.25	Sum of Wgt.	291
50%	4.5		Mean	4.609966
		Largest	Std. Dev.	.3434897
75%	4.87	5.5		
90%	5	5.56	Variance	.1179852
95%	5.25	5.62	Skewness	.7337865
99%	5.56	5.75	Kurtosis	2.863958

```
. summarize wageafter if state == "NJ", detail
```

#### wageAfter

	Percentiles	Smallest		
1%	5.05	5		
5%	5.05	5.05		
10%	5.05	5.05	Obs	291
25%	5.05	5.05	Sum of Wgt.	291
50%	5.05		Mean	5.081478
		Largest	Std. Dev.	.1056419
75%	5.05	5.5		
90%	5.15	5.67	Variance	.0111602
95%	5.25	5.67	Skewness	3.894742
99%	5.67	5.75	Kurtosis	18.89048

```
.
. * interquartile range
. tabstat wagebefore wageafter if state == "NJ", statistics(iqr) columns(statistics)

variable |      iqr
-----+-----
wagebefore | .6199999
wageafter  |      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 \$.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 `centile()` command can generate any quantiles by specifying the `centile()` option. 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 `0(10)100` sequence within the `centile()` option to create a sequence of numbers 0, 10, ..., 90, 100. The `0(10)100` sequence tells Stata to start at 0 and add 10 each time until it hits 100. We could alternatively specify the individual values with `centile(0 10 20 30 40 50 60 70 80 90 100)` and receive the same results. This command also provides us with confidence intervals, which we can ignore for now. Confidence intervals will be discussed in chapter 6.

```
. centile wagebefore if state == "NJ", centile(0(10)100)
                                         — Binom. Interp. —
                                         [95% Conf. Interval]
Variable |   Obs   Percentile    Centile
-----+-----+-----+-----+-----+-----+
wagebefore | 291       0     4.25     4.25     4.25*
                  10    4.25     4.25     4.25
                  20    4.25     4.25     4.25
                  30    4.25     4.25     4.25     4.393219
                  40    4.5      4.37     4.5
                  50    4.5      4.5      4.62
                  60    4.654    4.6      4.75
                  70    4.75    4.75     4.87
                  80      5     4.87     5
                  90      5      5      5.12
                 100    5.75    5.75     5.75*
```

* Lower (upper) confidence limit held at minimum (maximum) of sample					
. centile wageafter if state == "NJ", centile(0(10)100)					
Variable	Obs	Percentile	Centile	— Binom. Interp. —	
				[95% Conf. Interval]	
wageafter	291	0	5	5	5*
		10	5.05	5.05	5.05
		20	5.05	5.05	5.05
		30	5.05	5.05	5.05
		40	5.05	5.05	5.05
		50	5.05	5.05	5.05
		60	5.05	5.05	5.05
		70	5.05	5.05	5.05
		80	5.05	5.05	5.05
		90	5.19	5.05	5.25
		100	5.75	5.75	5.75*

\* Lower (upper) confidence limit held at minimum (maximum) of sample

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

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

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. For this calculation, we use the `egen` command introduced earlier in this chapter. As reviewed, the `egen` command offers extended functions to the `generate` command, including functions that enable us to calculate summary statistics for the entire data set or by subgroup.

```

. egen sqNJ = mean((fullpropaft - fullpropbef)^2) if state == "NJ"
(67 missing values generated)

. generate rmsNJ = sqrt(sqNJ)
(67 missing values generated)

.

. egen meanNJ = mean(fullpropaft - fullpropbef) if state == "NJ"
(67 missing values generated)

.

. summarize rmsNJ meanNJ

```

Variable	Obs	Mean	Std. Dev.	Min	Max
rmsNJ	291	.3014669	0	.3014669	.3014669
meanNJ	291	.0238747	0	.0238747	.0238747

On average, the absolute magnitude of change in the proportion of full-time employees, after the minimum wage was raised, is about  $.3$ . 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:

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

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 6, 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 5.

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 Stata, we can use the `summarize` command with the `detail` option to display both the standard deviation and variance. However, we can also use `tabstat` for a truncated and cleaner presentation of results for multiple variables, including by subgroup (states, in our example). We add the `long` and `nototal` options to, respectively, include the summary statistic labels and exclude calculations for the total sample.

		fullpr_e	fullpr_r
state	stats		
NJ	sd	.2304592	.2510016
	variance	.0531115	.0630018
PA	sd	.2397895	.2472422
	variance	.057499	.0611287

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 trials (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 Stata. 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. Stata 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.<sup>6</sup> A longitudinal study is one in which the same participants are

<sup>6</sup>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)

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 and overall high school GPA) continued through to the end of participants' high school attendance.

We will analyze just a portion of these data to investigate whether the small class sizes improved educational performance or not. The data file name is `STAR.dta`. 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. Label variable `classtype` according to the categories listed in table 2.6 (e.g., label 1 as `small`). 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; then add labels.
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. 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*. Use the

`if` qualifier to perform the percentile calculations within each class type. 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 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 creating a binary variable that categorizes white versus minority students (i.e., blacks and Hispanics), omitting the "other" category. Then compare the average reading and math test scores between white and minority students 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 regarding support on gay marriage.<sup>7</sup> 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 20 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 aimed to test the "contact hypothesis," which contends that outgroup hostility (toward gay people in this case) diminishes when people from different groups interact with one another. The data file is `gay.dta`. Table 2.7 presents the names and descriptions of the variables in this data set. Each observation of this data set is an individual's 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.

<sup>7</sup>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.

**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.

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). In chapter 4, we introduce *loops*, which will allow us to more efficiently perform calculations across waves.
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

**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

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.

### 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.<sup>8</sup> Each observation of the Stata data set `leaders.dta` 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

<sup>8</sup>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.

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? (Running the `return list` command after the `tabulate` command will return the total number of observations as well as the number of unique values.) 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. 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 3 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 called `warbefore`. Code the variable such that it is equal to 1 if a country is in either civil or international war during the 3 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 additional descriptive statistics in Stata.

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

**Table 3.1.** Afghanistan Survey Data.

Variable	Description
province	province where the respondent lives
district	district where the respondent lives
villageid	ID of the village where the respondent lives
age	age of the respondent
educyears	years of education of the respondent
employed	whether the respondent is employed
income	monthly income of the respondent (five levels)
violentexpISAF	whether the respondent experienced violence by ISAF
violentexpTaliban	whether the respondent experienced violence by the Taliban
listgroup	randomly assigned group for list experiment (control, ISAF, Taliban)
listresponse	response to the list experiment question (0–4)

A group of social scientists conducted a public opinion *survey* in southern Afghanistan, the heartland of the insurgency.<sup>1</sup> The survey was administered to a sample of 2,754 respondents between January and February 2011. The researchers noted that the participation rate was 89%. That is, they originally contacted 3,097 males and 343 of them refused to take the survey. Because local culture prohibited interviewers from talking to female citizens, all the respondents were 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). Once we set the working directory, we load the Stata file `afghan.dta` with the `use` command. The names and descriptions of the variables are given in table 3.1. We use the `summarize` command to provide numerical summaries of respondent age, education, and employment because they are continuous variables, and `tabulate` for the five-category income variable.

```
. cd measurement
. use afghan, clear
```

```
. * summaries for variables of interest
. summarize age educyears employed
```

<sup>1</sup>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.

Variable	Obs	Mean	Std. Dev.	Min	Max
age	2,754	32.39143	12.28949	15	80
educyears	2,754	4.002179	4.749902	0	18
employed	2,754	.5827887	.493188	0	1
. tabulate income					
	income	Freq.	Percent	Cum.	
	less than 2,000	457	17.58	17.58	
	2,001-10,000	1,420	54.62	72.19	
	10,001-20,000	616	23.69	95.88	
	20,001-30,000	93	3.58	99.46	
	over 30,000	14	0.54	100.00	
	Total	2,600	100.00		

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 US 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 `violentexpisaf` and `violentexptaliban` variables, which represent whether the respondents were harmed by the ISAF and the Taliban, respectively.

Using the `tabulate` command with the `cell` and `nofreq` options, we can create a cross-tabulation of proportions. Remember that the `nofreq` option provides proportions instead of raw counts and `cell` will calculate the proportions for each individual cell in the contingency table, totaling 100 as a whole.

.	*	cross-tabulation of proportions
.	tabulate	violentexpisaf violentexptaliban, cell nofreq
violentexp	violentexptaliban	
isaf	0	1
0	49.53	13.18
1	17.69	19.59
Total	67.23	32.77
		100.00

We see 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 Stata

Missing values are common in many types of data. For example, many developing countries lack certain official statistics such as the gross domestic product (GDP) or unemployment rate. In surveys, researchers often encounter nonresponse because either respondents refuse to answer some questions or they simply do not know the answer. In our data, the `tabulate` command ignored missing values, as if observations with missing values were not part of the data set. We can tell Stata to explicitly account for missing data when running this command. Adding the option `missing` after `tabulate` will include all data, even those with missing values.

<code>. tabulate violentexpisaf violentexptaliban, cell nofreq missing</code>				
violentexp isaf	violentexptaliban			Total
	0	1	.	
0	48.29	12.85	0.80	61.95
1	17.25	19.10	0.80	37.15
.	0.25	0.29	0.36	0.91
Total	65.80	32.24	1.96	100.00

We find that almost all respondents answered the victimization questions. Indeed, the nonresponse rates for these questions are less than 1% for the ISAF and less than 2% for the Taliban.<sup>2</sup> 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 were willing to answer questions about their experiences of violence.

As briefly mentioned in chapter 2, missing data are coded as a period (.) for numeric data and as an empty string ("") for string variables. Stata offers 26 extended missing values for numeric data. Following the alphabet in lowercase, these extended values range from `.a` to `.z`. This option allows users to make a distinction between different reasons for missingness (e.g., `.a` could mean nonresponse and `.b` could represent an administrative error). Missing values are treated as though they are a very large number, so care must be taken when using relational operations in data sets with missing values. For example, subsetting data in our sample to `income > 3` would include all observations with an income of greater than 10,000

<sup>2</sup> 3.6% did not respond to either question.

Afghani, but would also include those missing a value on income. As a result, our condition should be `income > 3 & income < .` to exclude the missing cases.

### 3.2.1 MISSINGS PACKAGE

The **missings** package can be quite useful when dealing with missing data in Stata. This package is available through the Statistical Software Components (SSC) archive so we can install it by typing `ssc install missings`. We briefly look at two of its commands.

The first command is `missings report`, which displays a table of missing values for the specified variables. The `missings` command works only with numeric variables. This command enables us to examine missingness across a multitude of variables, without having to run, for example, the `tabulate` or `codebook` commands individually for each. Here, we look at missingness on just the `violentexpisaf` and `violentexptaliban` variables. We see that 25 respondents are missing a response on whether they experienced violence by the ISAF and there are 54 missing values for the Taliban item.

```
. * number of observations missing values
. missings report violentexpisaf violentexptaliban

Checking missings in violentexpisaf violentexptaliban:
69 observations with missing values

+-----+
| # |
+-----+
| 25 |
| 54 |
+-----+
```

Another useful command is `missings dropobs`, which will drop observations missing data on the specified variables. We may want to subset our data to just those respondents who answered the two questions regarding experiences of violence. When the `missings dropobs` command contains more than one variable, only observations missing values on each of those variables will be dropped. An observation with at least one observed value across the listed variables will remain in the data set. Because our previous commands changed our data set, we either have to save the data in memory before running `missings dropobs` or use the `force` option so Stata will process our request without generating an error (i.e., forcing the command through). We create a cross-tabulation using `tabulate` to see that observations with at least one nonmissing value on the violence variables remain in the data set, but observations missing data on both variables have been removed.

```
. missings dropobs violentexpisaf violentexptaliban, force

Checking missings in violentexpisaf violentexptaliban:
69 observations with missing values
(10 observations deleted)
```

. tabulate violentexp isaf violentexptaliban, missing				
violentexp isaf	violentexptaliban			Total
	0	1	.	
0	1,330	354	22	1,706
1	475	526	22	1,023
.	7	8	0	15
Total	1,812	888	44	2,744

The **missings** package offers a similar subcommand `dropvars` that will drop variables that have no valid values and only take up space. The `missings` command can also be used to examine missingness within subgroups. For example, running `bysort income: missings table` will show the number of missing values for each category of income. In addition, by using the `missings` tag command with the `generate()` option, we can create a new variable that flags observations missing values, including on specific variables. Below, we create a variable called `miss_income` that tags respondents who are missing income data. Readers can type `help missings` to see additional subcommands and options available in the **missings** package.

. missings tag income, generate(miss_income)
Checking missings in income:
149 observations with missing values
. tabulate miss_income
miss_income   Freq. Percent Cum.
0   2,595 94.57 94.57
1   149 5.43 100.00
Total   2,744 100.00

### 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 Stata. Please note that the graphs we present have been edited from the original Stata format. While the graphs do not look exactly like those produced in Stata, the core elements remain the same.

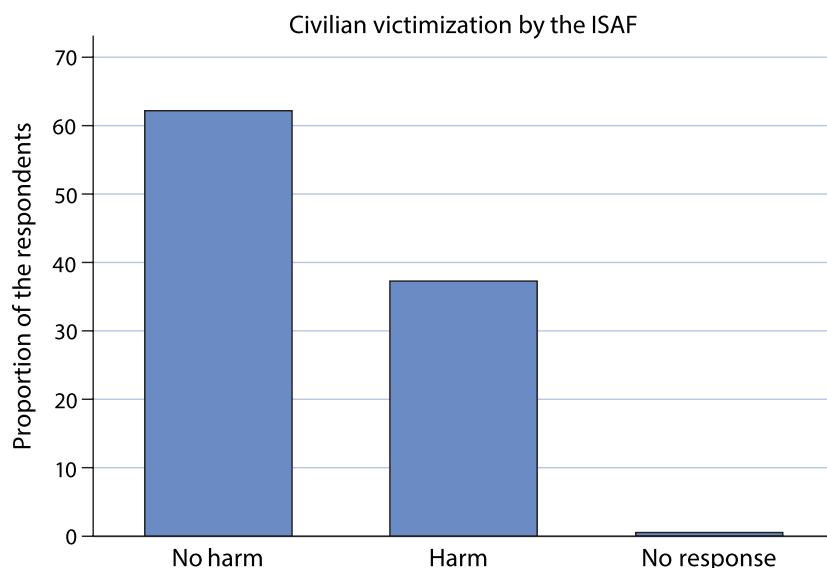
### 3.3.1 BAR PLOT

To summarize the distribution of a *factor variable* or *factorial variable* with several categories, a simple table with counts or proportions using the `tabulate` command is often sufficient. But it is also possible to use a *bar plot* to visualize the distribution. In Stata, we can use the `graph bar` command to display a bar plot in a separate graphical window. In this example, the height of the bars represents the proportion of respondents in each response category. We first run the `tabulate` command for comparison. For all commands, we include the `missing` option to display the proportion of nonresponse in our sample.

```
. * victimization by ISAF
. tabulate violentexpisaf, missing

violentexpisaf
      saf |   Freq.    Percent     Cum.
      0 | 1,706     62.17    62.17
      1 | 1,023     37.28    99.45
      . |    15      0.55    100.00
      Total | 2,744     100.00

.
. * make bar plots by specifying a certain range for y-axis
. graph bar, over(violentexpisaf, relabel(1 "No harm" 2 "Harm" ///
> 3 "No response")) missing ylabel(0(10)70) ///
> ytitle("Proportion of the respondents") ///
> title("Civilian victimization by the ISAF")
```



```

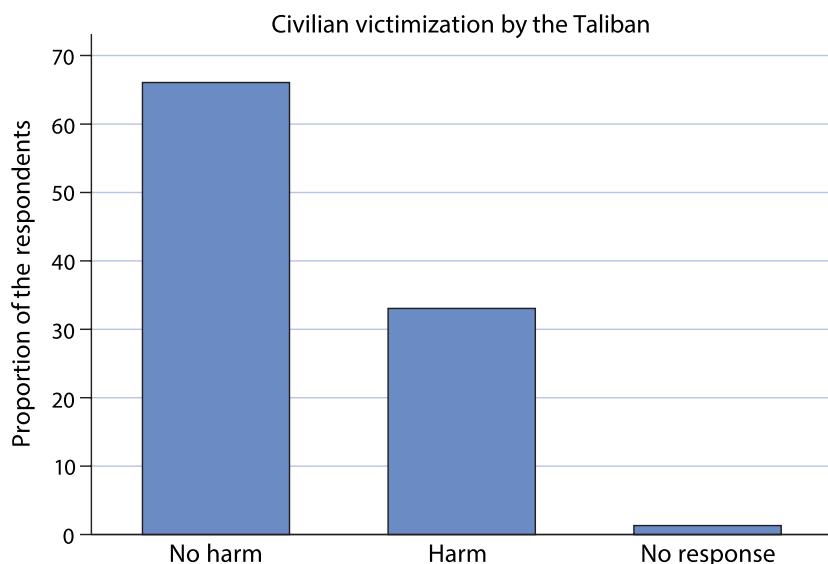
. * repeat the same for the victimization by Taliban
. tabulate violentexptaliban, missing

violentexpt
      aliban |   Freq.    Percent     Cum.
  _____|_____|_____|_____
           0 | 1,812     66.03    66.03
           1 |  888     32.36   98.40
          . |   44      1.60  100.00
  _____|_____|_____|_____
           Total | 2,744    100.00
.
```

```

. graph bar, over(violentexptaliban, relabel(1 "No harm" 2 "Harm" ///
> 3 "No response")) missing ylabel(0(10)70) ///
> ytitle("Proportion of the respondents") ///
> title("Civilian victimization by the Taliban")

```



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. The above syntax illustrates the use of several arguments common to other plot functions in Stata and are summarized here:

- **title ()**: A character string, i.e., a series of characters in double quotes, for the main title of the plot.
- **ytitle (), xlabel ()**: Character strings for labeling the vertical axis (i.e., *y*-axis) and the horizontal axis (i.e., *x*-axis), respectively. (Stata will automatically set these arguments to the default labels if left unspecified.)
- **ylabel (), xlabel ()**: Numeric ranges specifying the interval for the *y*-axis and *x*-axis, respectively. (Stata will automatically set these arguments if left unspecified.)

- `over()`: Tells Stata that the data should be graphed across the categories of the variable that follows this option.
- `relabel()`: Overwrites value labels for the graphed categorical variable, providing bar labels on the  $x$ -axis. This is a suboption to `over`.
- `missing`: Keeps observations that are missing values in the graph. (Stata drops these observations if `missing` is not specified.)

It is important to note that a comma typically follows the full graph command, prior to adding most of these customizations. Stata offers many other options for tailoring the display of graphs, including the specification of colors, line types, marker symbols, legend options, grid lines, and so on. A few examples of additional features that are commonly used include:

- `color()`: Changes the color of various elements in the graph. This argument can be used in many commands including `graph bar`, `histogram`, `twoway line`, and `twoway scatter`. In the latter two commands, the line color is specified by `lcolor()`. Type `help colorstyle` to view Stata's list of built-in color names.
- `lpattern()`: Specifies the type of line to be drawn using either a name or a formula. Users can specify `solid` (default) for solid lines, `dash` for dashed lines, `dot` for dotted lines, `dash_dot` for dotted and dashed lines, `longdash` for long dashed lines, or they can create their own formula of symbols within quotations (e.g., `lpattern ("-. .")`).
- `msymbol()`: Selects shape of marker symbols used in graphs. This argument is used mainly with the `twoway scatter` command. Available symbols can be viewed by typing `help symbolstyle`. The `msize()` and `mcolor()` options similarly allow one to adjust marker size and color, respectively.
- `scheme()`: Specifies the graph's color scheme. The `s2mono` scheme produces a monochrome graph. Other options can be seen by typing `help scheme`. (Stata uses its default scheme 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 of [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

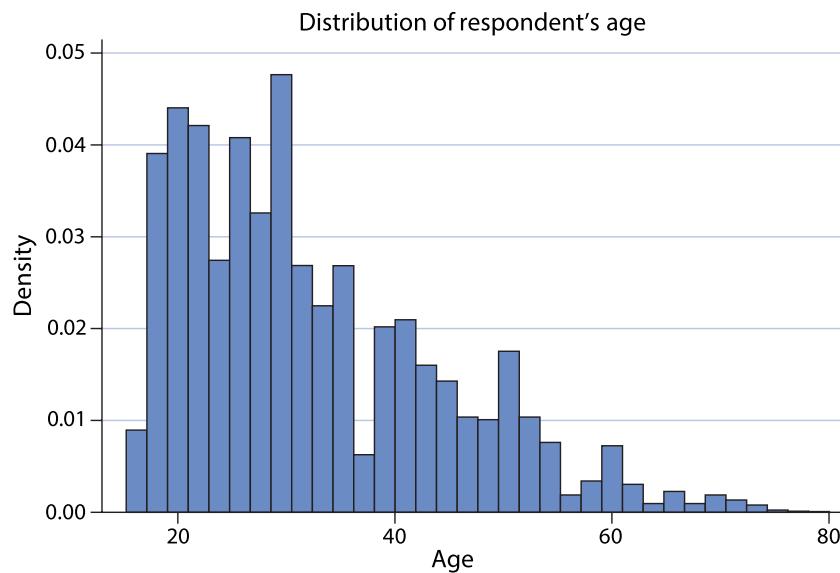
$$\text{density} = \frac{\text{proportion of observations in the bin}}{\text{width of the bin}}.$$

We often care more about the variable's distribution as shown by the relationship of the different bins' densities to one another within a histogram, rather than the exact value of

each density. We can therefore think of histograms as rectangular approximations of the distribution.

To create histograms in Stata, we use the `histogram` command. The default for this command is to plot the density as the height of each bin rather than the frequency, i.e., counts. 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. To show the frequency rather than density, we would add the `frequency` option to our command. Though the edited graph looks slightly different, we customize the graph in our Stata code so that the bars are outlined in black (`color(black)`) and have no fill (`fcolor(none)`).

```
. histogram age, xtitle("Age") ///
>           title("Distribution of respondent's age") ///
>           fcolor(none) color(black)
(bin=34, start=15, width=1.9117647)
```



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, `educyears`. Instead of letting Stata automatically choose the width of bins, as we did for the age variable, we now specify exactly how the bins are created using the `discrete` option. This option centers each bin around an integer value, i.e., 0, 1, 2, . . . , 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. Note that we can also specify the number of bins for the histogram using the `bin()` option where the desired number of bins goes in parentheses.

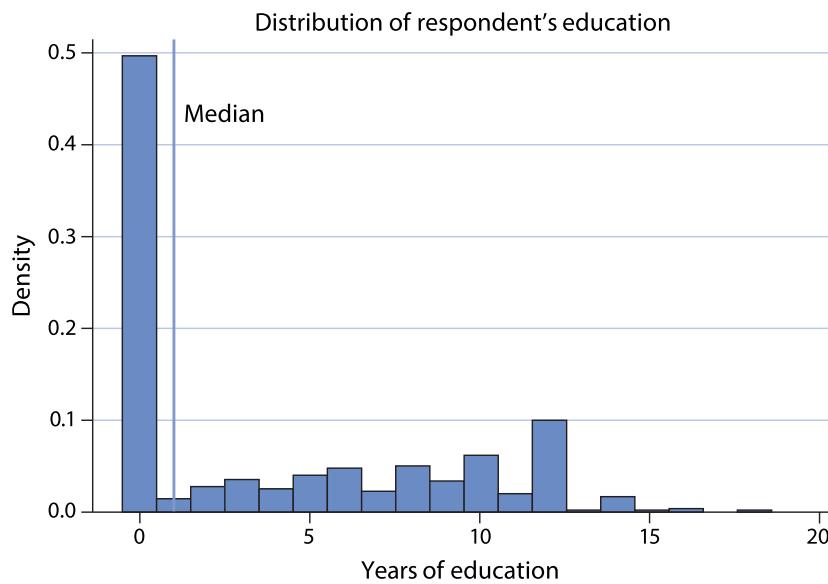
Remember from section 2.4.2 that Stata stores the results from the `summarize` command, and we can access the median stored in `r(p50)` when adding the `detail` option. We add the *median* value to our graph as a vertical line using the `xline()` option and surround the stored result `r(p50)` by a backtick and apostrophe. To add a horizontal line, we would use the `yline()` option. We can place multiple vertical or horizontal lines on the same graph by specifying multiple *x* or *y* values in the `xline()` or `yline()` options. Stata also allows us to place text anywhere on the existing plot with the `text` option, respectively specifying the *y*- and *x*-coordinates for where the text should be placed, followed by the text itself in quotations.

```
. * add a vertical line representing median
. summarize educyears, detail

      educyears

+-----+
Percentiles      Smallest
1%              0              0
5%              0              0
10%             0              0      Obs            2,744
25%             0              0      Sum of Wgt.   2,744
50%             1
                           Mean       4.012755
                           Largest
                           Std. Dev.  4.753438
75%             8              18
90%            12              18      Variance     22.59517
95%            12              18      Skewness     .6986662
99%            14              18      Kurtosis    1.980407

. histogram educyears, discrete xline(`r(p50)') ///
>           xtitle("Years of education") ///
>           title("Distribution of respondent's education") ///
>           text(.45 2.5 "median") fcolor(none) color(black)
(start=0, width=1)
```

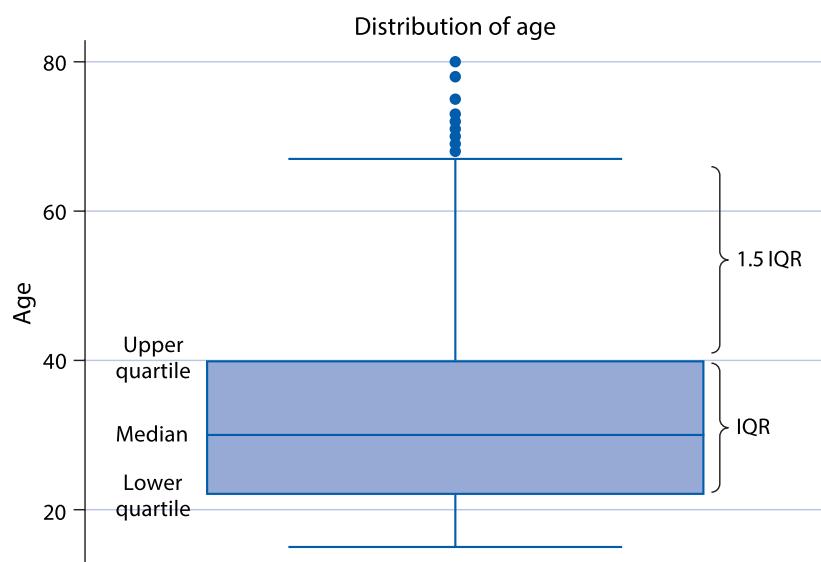


The histogram for the years of education variable clearly shows that the education level of these respondents is extremely low. Indeed, almost half of them have never attended school.

### 3.3.3 BOX PLOT

The *box plot* represents another way to visualize the distribution 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 interquartile range (IQR) all together as a single object. To make box plots in Stata, we use the `graph box` command followed by the variable of interest. Again, we use the `age` variable as an example.

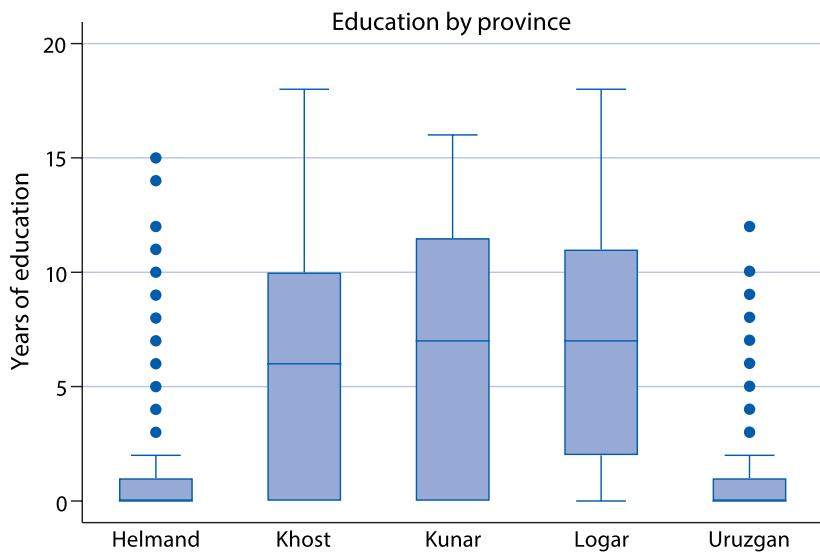
```
. graph box age, ytitle("Age") title("Distribution of age")
```



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, two 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 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 closed circles. In this plot, the section of the vertical 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 vertical line will end at the minimum (maximum) value. The absence of closed 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 distribution. One of the main advantages of box plots is that it allows us to compare multiple distributions in a more compact manner than histograms, as the next example shows. Using the `graph box` command, we can create a box plot for a different group of observations where the groups are defined by another variable. This is done by using the `over()` option in Stata. In the current context, `graph box y, over(x)` creates box plots for variable `y` for different groups defined by `x`. As an illustration, we plot the distribution of the years of education variable by province.

```
. graph box educyears, over(province) ytitle("Years of education") ///
>           title("Education by province")
```



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 confirmed

by computing the proportion of affirmative answers to the corresponding question, for each province. We use the `tabstat` command introduced in chapter 2 to compute the means across provinces for multiple variables.

```
. tabstat violentexptaliban violentexpisaf, by(province) statistics(mean)

Summary statistics: mean
    by categories of: province

  province |   violen_n   violen_f
  Helmand | .504222   .5410226
  Khost   | .2332268   .2424242
  Kunar   | .3030303   .3989899
  Logar   | .0802469   .1440329
  Uruzgan | .4545455   .4960422
  Total   | .3288889   .3748626
```

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 Stata. The easiest way is to use the menus. A new window will open each time Stata creates a graph using any of the plotting functions. To save an image of the plot, click `File` → `Save` or `File` → `Save as`. We can then choose the type of file to save the graph as. Options include `.pdf`, `.eps`, `.png`, `.tiff`, or Stata's own `.gph` format.

We can also save or print a graph with the `graph save` command after running a `graph` command. Alternatively, we could specify the saving option within the `graph` command. For example, the syntax that follows shows both ways to save the box plots we just created in the working directory. The `replace` option saves over a file if one with the same name already exists. If no format is specified, Stata will save the graph as `.gph` by default.

```
. * saving or printing a graph
. graph save educyears.pdf, replace
(file educyears.pdf saved)

. graph box educyears, over(province) ytitle("Years of education") ///
>           title("Education by province") ///
```

```
>           saving(educyears, replace)
(file educyears.gph saved)
```

Finally, we can save a copy of the graph in memory rather than to disk (i.e., on our local drive) by using the `name()` option. Without either the `saving()` or `name()` options, Stata refers to the just-created graph as `Graph` and assumes that you want to replace the graph in memory. The `name()` option saves a temporary version of our graph that will be removed when Stata is closed. The `graph dir` command shows a list of the graphs currently stored in memory. The `graph display` command will redraw graphs with alterations and `graph use` opens a graph saved to disk.

```
. * example alteration with display and opening a saved graph
. graph display, scheme(s1mono)
. graph use educyears.gph
```

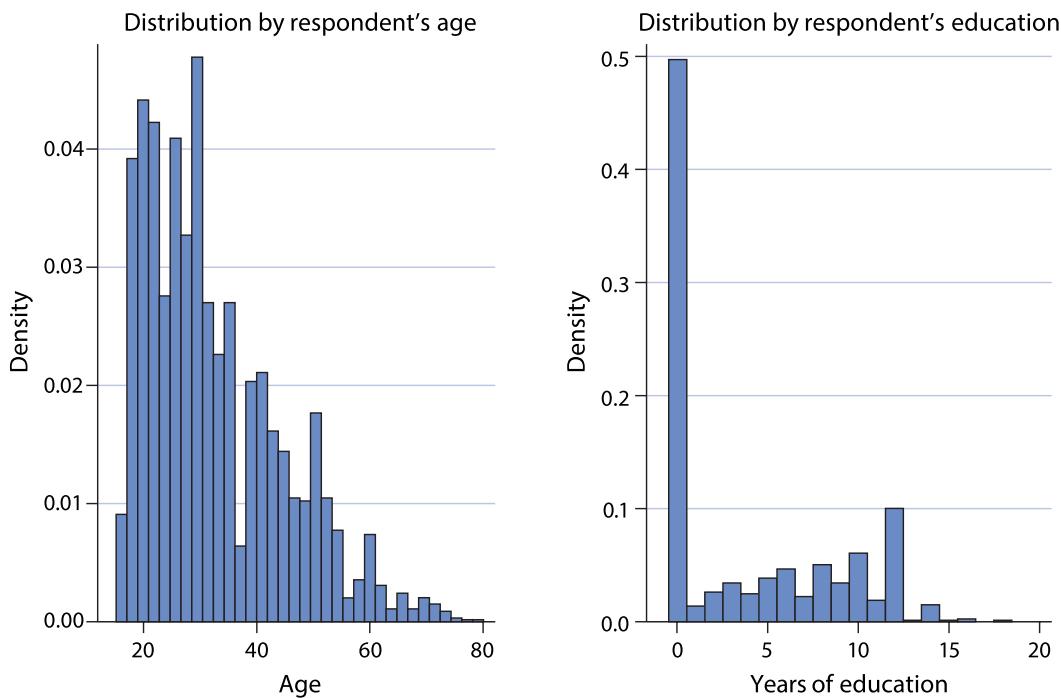
In many cases, we want to compare multiple plots by printing them next to each other in a single figure file. To do this, we use the Stata command `graph combine` after we have created our graphs. This combines the graphs we specify into a new figure. We can indicate the number of rows or columns we want with the `rows(X)` or `cols(X)` options. The default in Stata is to stack two graphs side-by-side before adding a second row. If we want two graphs to be on separate rows, we would add the `rows(2)` option. Note that the `graph combine` command also takes many other arguments that allow users to control graphics in Stata. For example, if we want our combined graphs to have the same *y*-axis scale, we would specify the `ycommon` option. We would use `xcommon` to put the graphs on the same *x*-axis scale. We could also add an overall title to the combined graph using the `title()` option.

Executing the following code block all at once creates the two histograms we made earlier in this chapter and combines them side by side. We give each of the histograms a name using the `name()` option. This stores a copy of each graph in current memory. If there is already a graph with the name in memory, Stata will not overwrite it unless we use the `replace` suboption. We can then refer to each of the graphs by name when using the `graph combine` command.

```
. * combining graphs
. histogram age, ylabel(0(.01).04) ///
>           xtitle("Age") title("Distribution of respondent's age") ///
>           fcolor(none) color(black) name(age, replace)
(bin=34, start=15, width=1.9117647)

. histogram educyears, discrete ///
>           xtitle("Years of education") ///
>           title("Distribution of respondent's education") ///
>           fcolor(none) color(black) name(educ, replace)
(start=0, width=1)

. graph combine age educ
```



### 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 2,754 respondents was used to infer the experiences and attitudes of approximately 15 million civilians. In the United States, a sample of just about 1,000 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 6 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. To illustrate, 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,

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: *Chicago Tribune*

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.

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.

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.

**Table 3.2.** Afghanistan Village Data.

Variable	Description
villagesurveyed	whether a village is sampled for survey
altitude	altitude of the village
population	population of the village

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.dta` 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 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  $e = 2.7182\dots$ , 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 `villagesurveyed` indicates whether each village in the data is (randomly) sampled and surveyed; 1 indicates yes and 0 no. As explained, we take the natural logarithmic transformation for the population variable using the

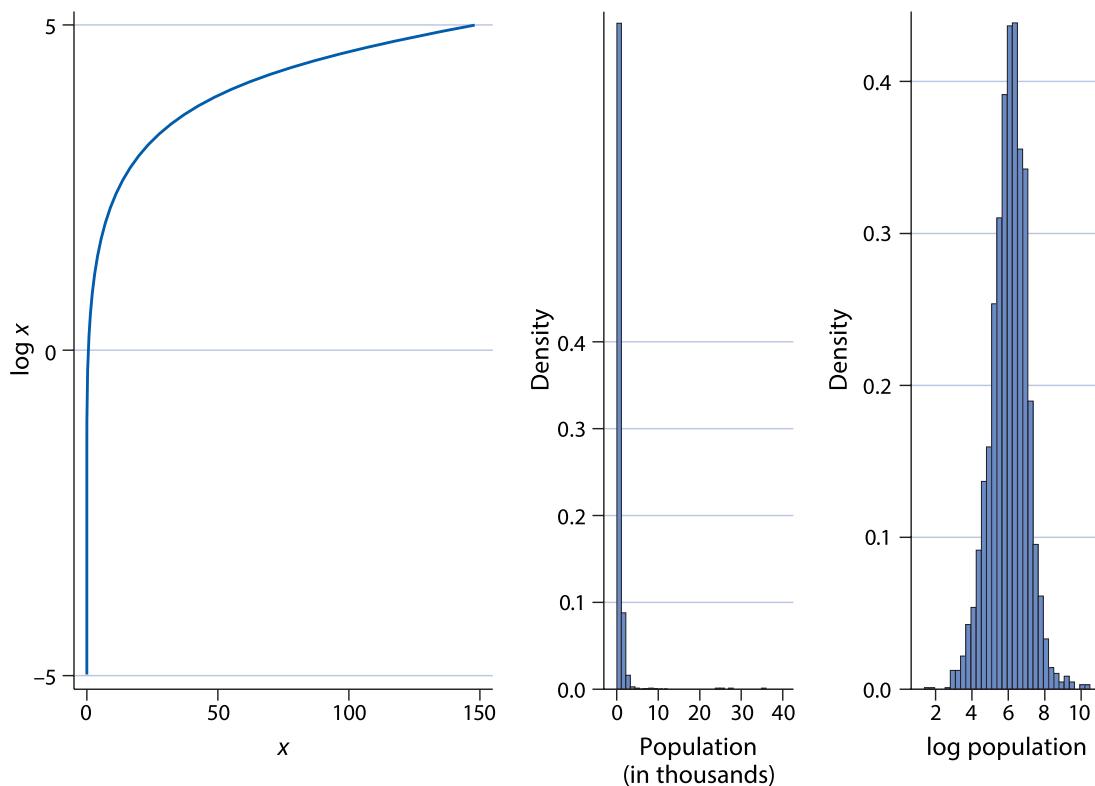


Figure 3.2. The Natural Logarithm. The left plot shows the natural logarithm  $\log_e x$  where  $x$  is a positive number and  $e = 2.7182\dots$  is Euler's number. The remaining plots display the histograms for the population of Afghan villages on the original scale (in thousands) and the natural logarithmic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

`log()` function. By default, Stata uses  $e$  as its base, though it is possible to specify a base of 10 with the `log10()` function. Note that the exponential function in Stata is given by `exp()`.

```
. * load village data
. use afghan-village, clear

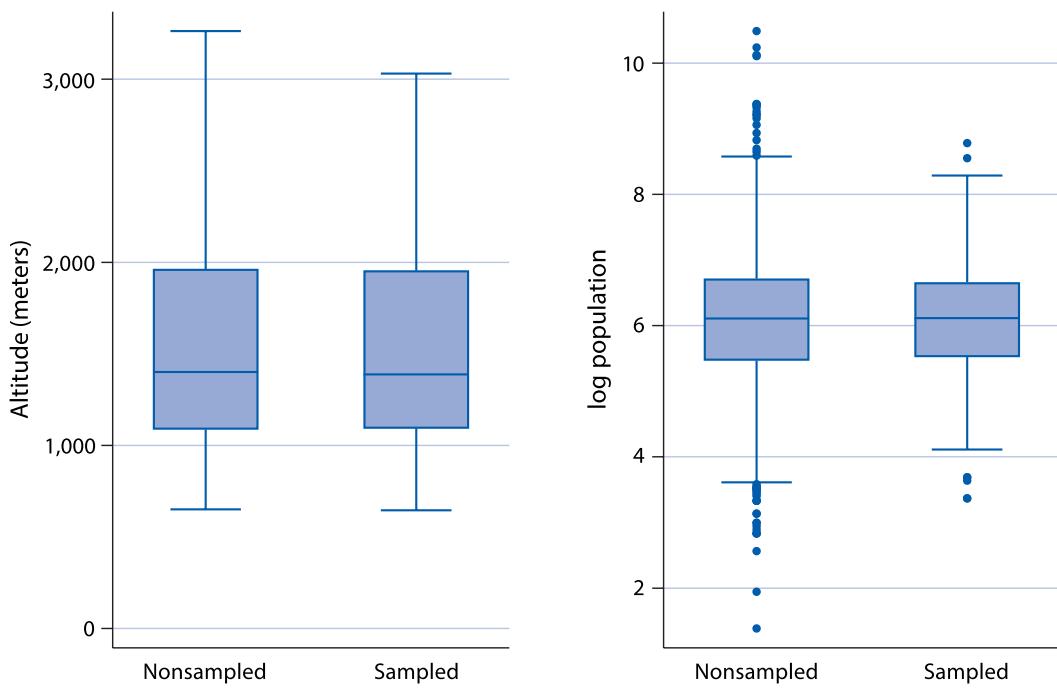
. * box plots for altitude and log population
. graph box altitude, over(villagesurveyed, relabel(1 "Nonsampled" 2 "Sampled")) ///
>     ytitle("Altitude (meters)") name(alt, replace)

. generate logpop = log(population)

. graph box logpop, over(villagesurveyed, relabel(1 "Nonsampled" 2 "Sampled")) ///
>     ytitle("log population") name(pop, replace)

. graph combine alt pop

.
```



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 2,754 out of 3,097 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. We saw in section 3.2 that in the Afghanistan survey, the income variable had a nonresponse rate of approximately 5.6%. 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. Combining the `generate` command with the `missing` function allows us to create a binary variable that indicates whether

an observation is missing a value on the variable contained within the parentheses. We can then view the nonresponse rate using `tabstat`.

```
. use afghan, clear
.
. * create binary variable flagging missing values
. generate missvioltaliban = missing(violentexptaliban)
. generate missviolisaf = missing(violentexpisaf)
.
. * view nonresponse rate, by province
. tabstat missvioltaliban missviolisaf, by(province) statistics(mean)

Summary statistics: mean
by categories of: province

province | missvi_n missvi_f
-----|-----
Helmand | .0304094 .0163743
Khost | .0063492 .0047619
Kunar | 0 0
Logar | 0 0
Uruzgan | .0620155 .0206718
-----|-----
Total | .0196078 .0090777
```

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 non-response** 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 for which 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 a 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 policymakers 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 items 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 `listgroup` 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 `listresponse`, which represents the item count reported by each respondent. As before, we store each group mean in a scalar, which will hold the values without adding anything to our data set.

```
. summarize listresponse if listgroup == "ISAF"
```

Variable	Obs	Mean	Std. Dev.	Min	Max
listresponse	918	1.568627	1.085834	0	4

```
. scalar isaf = r(mean)
. summarize listresponse if listgroup == "control"
Variable      Obs       Mean     Std. Dev.      Min      Max
listresponse   918     1.519608    1.04696      0       3
. scalar control = r(mean)
. display isaf - control
.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.

```
. tabulate listresponse listgroup
listrespon      listgroup
      se          ISAF    control    taliban      Total
      0           174     188        0       362
      1           278     265     433       976
      2           260     265     287       812
      3           182     200     198       580
      4            24       0        0        24
      Total       918     918     918      2,754
```

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.

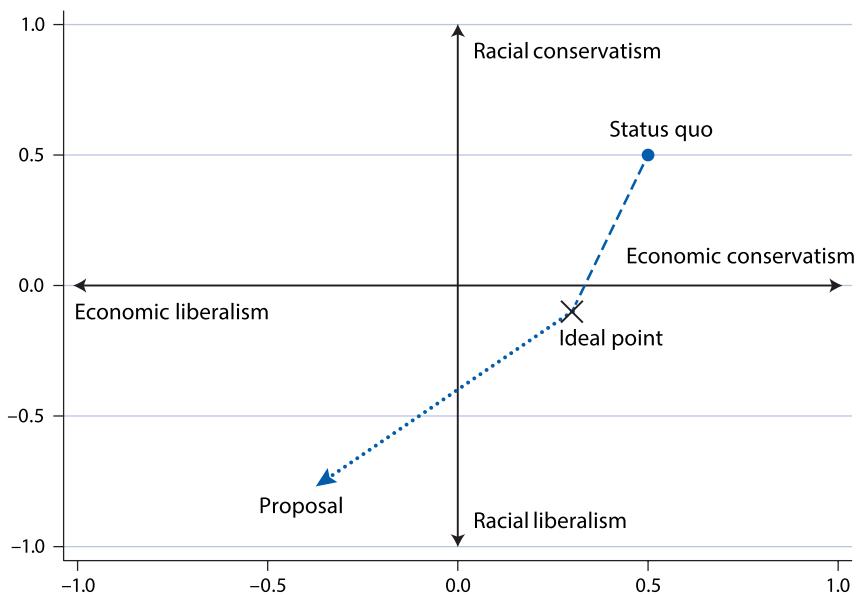


Figure 3.3. An Illustration for the Spatial Voting Model of Legislative Ideology.

### 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.<sup>3</sup>

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 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 circle) 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, an unanimously accepted or rejected proposal provides no information about legislators' ideological orientations.

<sup>3</sup>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.

**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)

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 (conservative). The Stata file `congress.dta` 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 SCATTERPLOT

Using the `scatter` command, we create a *scatterplot*, which plots one variable against another in order to visualize their relationship. The syntax for this function is `scatter y x`, where `x` and `y` are the arguments for the 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. Using this axis information, we create scatterplots of ideal points for the 80th and 112th Congresses.

To plot Democrats and Republicans separately, we use the `if` qualifier to run one scatter command when `party == "Democrat"` and one when `party == "Republican"`. The two `scatter` commands are separated by double bars `||`, which tells Stata to produce separate graphs. We can also use `||` to plot different types of graphs on the same axes, such as combining line and scatter graphs. We specify different plotting symbols for the two parties. We use the default of solid circles for Democrats and `msymbol(T)` graphs solid triangles

for Republicans. Other marker shapes include D for diamonds and S for squares. For more options, see `help symbolstyle`. Adding `legend(off)` suppresses the legend.

```

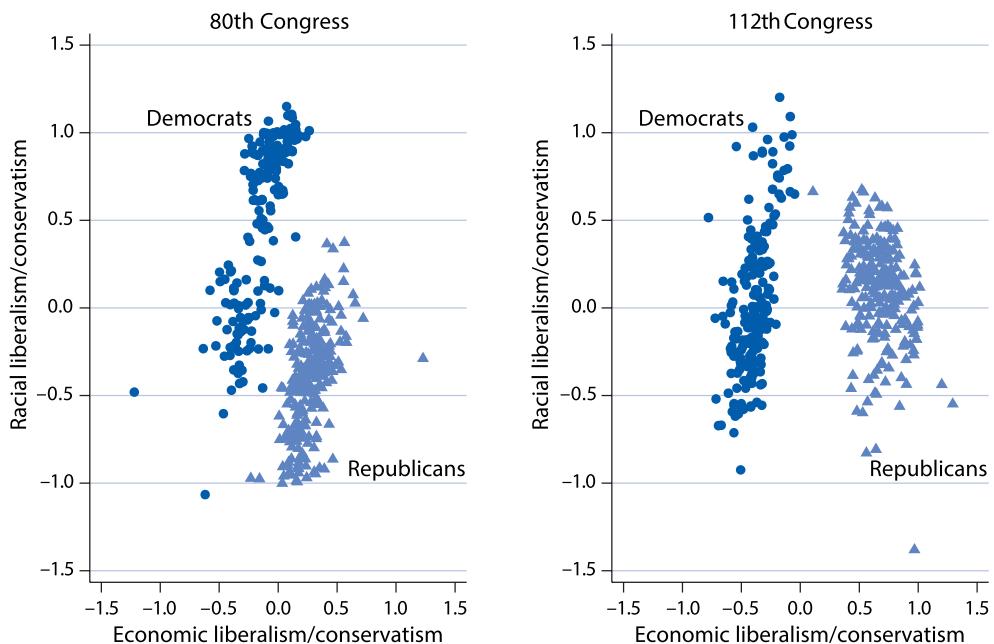
. * load the data
. use congress, clear

. * scatterplots for the 80th and 112th Congresses
. scatter dwnom2 dwnom1 if party == "Democrat" & congress == 80 || ///
>     scatter dwnom2 dwnom1 if party == "Republican" & congress == 80, ///
>     xlabel(-1.5(.5)1.5) ylabel(-1.5(.5)1.5) ///
>     xtitle("Economic liberalism/conservatism") ///
>     ytitle("Racial liberalism/conservatism") title("80th Congress") ///
>     text(1 -0.75 "Democrats") text(-1 1 "Republicans") ///
>     msymbol(T) mcolor(red) legend(off) name(congress80, replace)

. scatter dwnom2 dwnom1 if party == "Democrat" & congress == 112 || ///
>     scatter dwnom2 dwnom1 if party == "Republican" & congress == 112, ///
>     xlabel(-1.5(.5)1.5) ylabel(-1.5(.5)1.5) ///
>     xtitle("Economic liberalism/conservatism") ///
>     ytitle("Racial liberalism/conservatism") title("112th Congress") ///
>     text(1 -1 "Democrats") text(-1 1 "Republicans") ///
>     msymbol(T) mcolor(red) legend(off) name(congress112, replace)

. graph combine congress80 congress112

```



The plots (see also page C1) 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.

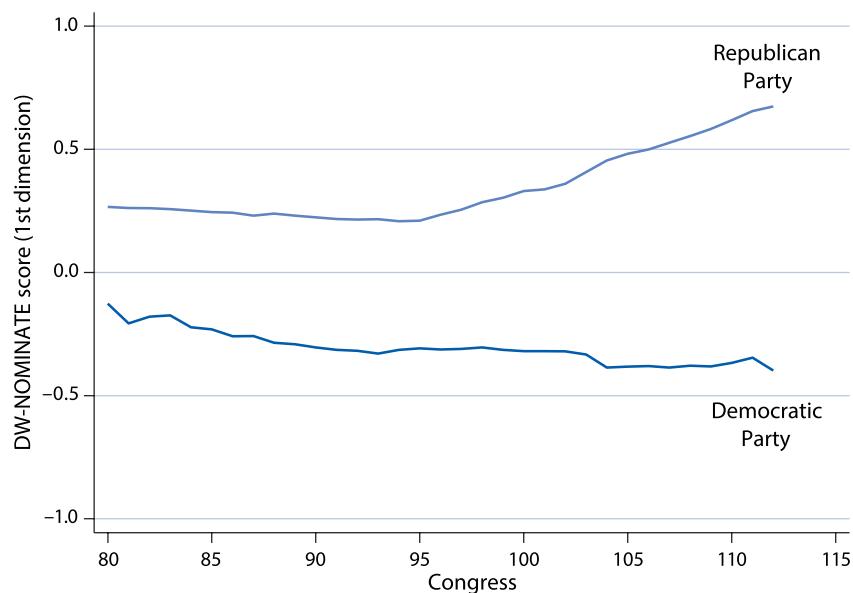
Next, we compute the median legislator, based on the DW-NOMINATE first dimension score, separately for the Democratic and Republican Parties and for each Congress. These party median ideal points represent the center of each party in the economic liberalism/conservatism dimension.

We first store the DW-NOMINATE first dimension scores separately for Democrats and Republicans with the `generate` command. The `bysort` command together with the `egen` command allows us to calculate each party's median score for each Congress. Remember from the exercises in chapter 1 that `bysort` tells Stata to perform the calculation following the colon separately for each category of the variable preceding the colon (in this case, for each party within each Congress). The `by` command alone requires that the data are sorted by the grouping variable, so `bysort` provides a more efficient command to ensure the data are in the necessary order before the group-based calculations can be made.

```
. * party median for each congress
. bysort congress party: egen partymedian = median(dwnom1)
```

Finally, using the `line` command, we create a *time-series plot* where each party median is displayed for each Congress. 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.

```
. line partymedian congress if party == "Democrat" || ///
>     line partymedian congress if party == "Republican", ///
>     xlabel(80(5)115) ylabel(-1(.5)1) ///
>     xtitle("Congress") ytitle("DW-NOMINATE score (1st dimension)") ///
>     text(-.6 110 "Democratic" "Party") text(.85 110 "Republican" "Party") ///
>     legend(off)
```



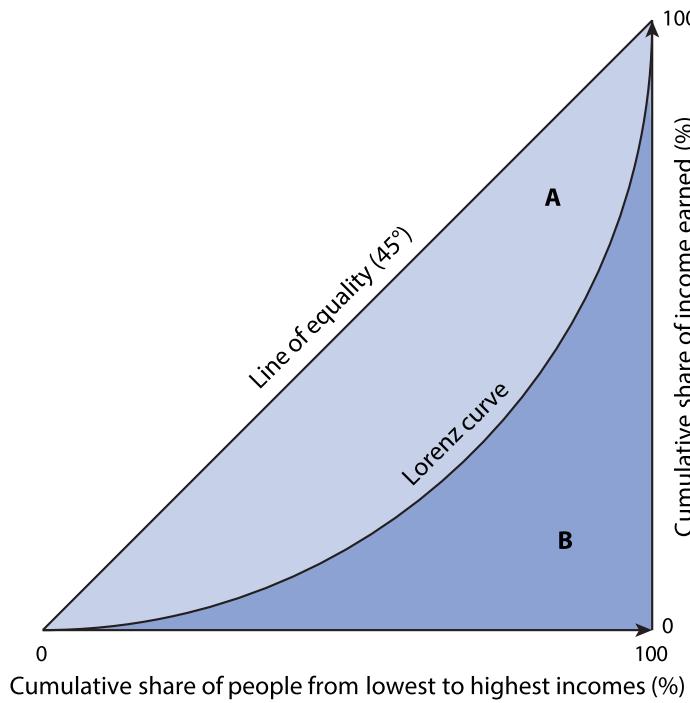


Figure 3.4. Gini Coefficient and Lorenz Curve. *Source:* Wikipedia ([http://en.wikipedia.org/wiki/Gini\\_coefficient](http://en.wikipedia.org/wiki/Gini_coefficient))

Note that in the `text()` option, we place double quotations around both "Democratic" and "Party" to move "Party" to a new line. The plot clearly shows that the ideological centers of the two parties diverge over time. The Democratic Party has become more liberal while the Republican Party has increasingly moved in a more conservative direction in recent years. Many scholars refer to this phenomenon as *political polarization*.

A **scatterplot** 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.

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,

$$\begin{aligned} \text{Gini coefficient} &= \frac{\text{area between the line of equality and the Lorenz curve}}{\text{area under the line of equality}} \\ &= \frac{\text{area A in Figure 3.4}}{\text{area A + area B in Figure 3.4}}. \end{aligned}$$

The formula implies 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. Our first plot will show the partisan gap, i.e., the difference between the two party medians, over time. The second time-series plot will display the Gini coefficient during the same time period. The Stata data file `USGini.dta` contains the Gini coefficient from 1947 to 2013 (see table 3.4). Before creating these plots, we must first merge the two data sets.

The `congress` data set should still be in our working memory. We store the median results for each party in new variables before using the `collapse` command to aggregate our data.

```
. generate demmedian = partymedian if party == "Democrat"
(6,420 missing values generated)

. generate repmedian = partymedian if party == "Republican"
(8,151 missing values generated)

. collapse demmedian repmedian, by(congress)
```

As reviewed in section 1.3.6, merging data requires that at least one variable is shared by both data sets. Consequently, we create a year variable that corresponds to each congress to act as our shared identifier. The `fill()` function with the `ege`n command allows us to generate a variable that contains a specific pattern or sequence of values. Within the parentheses of the `fill()` function, we indicate the starting point of our sequence (1948), end point (2012), and the increment by which the numbers should be increased (2).

**Table 3.4.** US Gini Coefficient Data.

Variable	Description
year	year
gini	US Gini coefficient

```
. * create year variable
. egen year = fill(1948(2)2012)
```

Within each data set, there is only a single observation for each year, so we use the `1 : 1` option with `merge`.

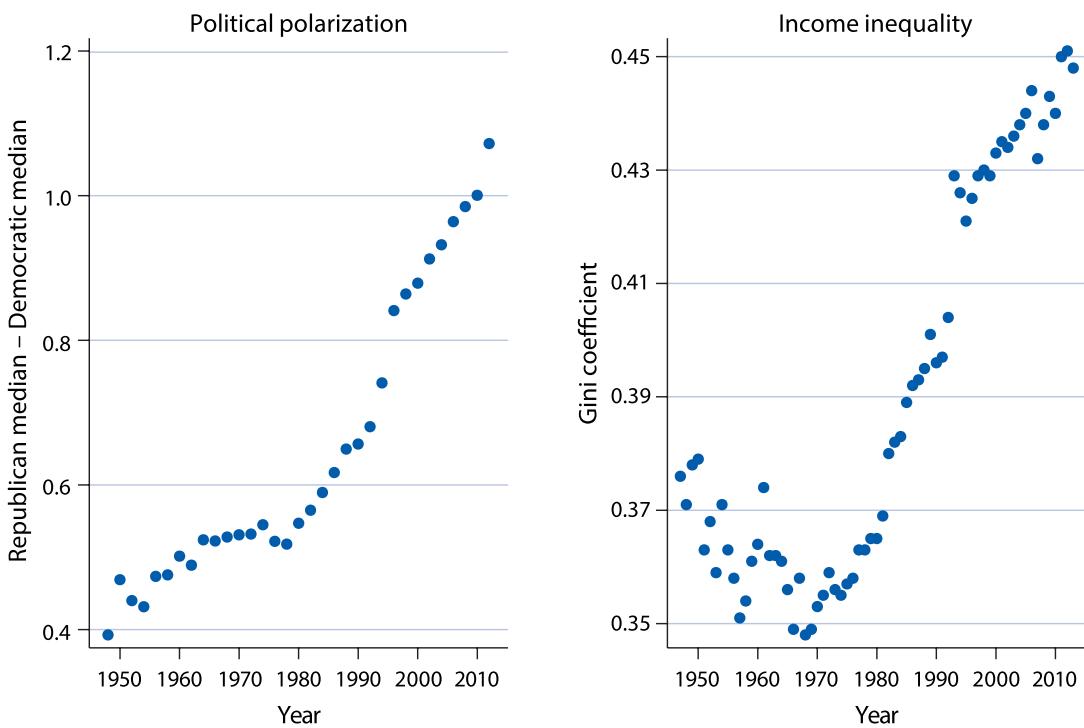
```
. merge 1:1 year using USGini
      Result          # of obs.
      not matched      34
      from master       0  (_merge==1)
      from using        34  (_merge==2)
      matched           33  (_merge==3)
```

We can see that 33 of the cases (or years) were successfully matched between the two data sets. Thirty-four cases from the using data set were not matched. This is because the years included in the `USGini` data set extend beyond those in the `congress` data set.

To graph each plot, we first generate a polarization variable called `polar`. This measure is the difference between the median Republican and median Democratic DW-NOMINATE scores on the first dimension.

```
. generate polar = repmedian - demmedian
(34 missing values generated)

.
. * time-series plots for partisan difference and Gini coefficient
. scatter polar year, xlabel(1950(10)2010) ///
>     xtitle("Year") ytitle("Republican median - Democratic median") ///
>     title("Political polarization") msymbol(Oh) name(polar, replace)
. scatter gini year, xlabel(1950(10)2010) ylabel(0.35(.02)0.45) ///
>     xtitle("Year") ytitle("Gini coefficient") ///
>     title("Income inequality") msymbol(Oh) name(gini, replace)
. graph combine polar gini
```



We notice that both political polarization and income inequality have been steadily increasing in the United States. 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  $i$ th observation of variable  $x$  is defined as

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

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:

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

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  $i$ th observation of a variable  $x$  measures the number of standard deviations an observation is above or below the mean. It is defined as

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

where  $\bar{x}$  and  $S_x$  are the mean and standard deviation of  $x$ . 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:

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

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. As an 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, likewise, 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.

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

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

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 Stata, the correlation can be calculated using the `correlate` command. To illustrate, we can now calculate the correlation between the Gini coefficient and the measure of political polarization. Because each US congressional session lasts 2 years, the correlation with the Gini coefficient is calculated only for the merged year of each session.

```
. correlate gini polar
(obs=33)
```

	gini	polar
gini	1.0000	
polar	0.9418	1.0000

The correlation is positive and quite high, indicating that political polarization and income inequality move in a similar direction. Again, 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

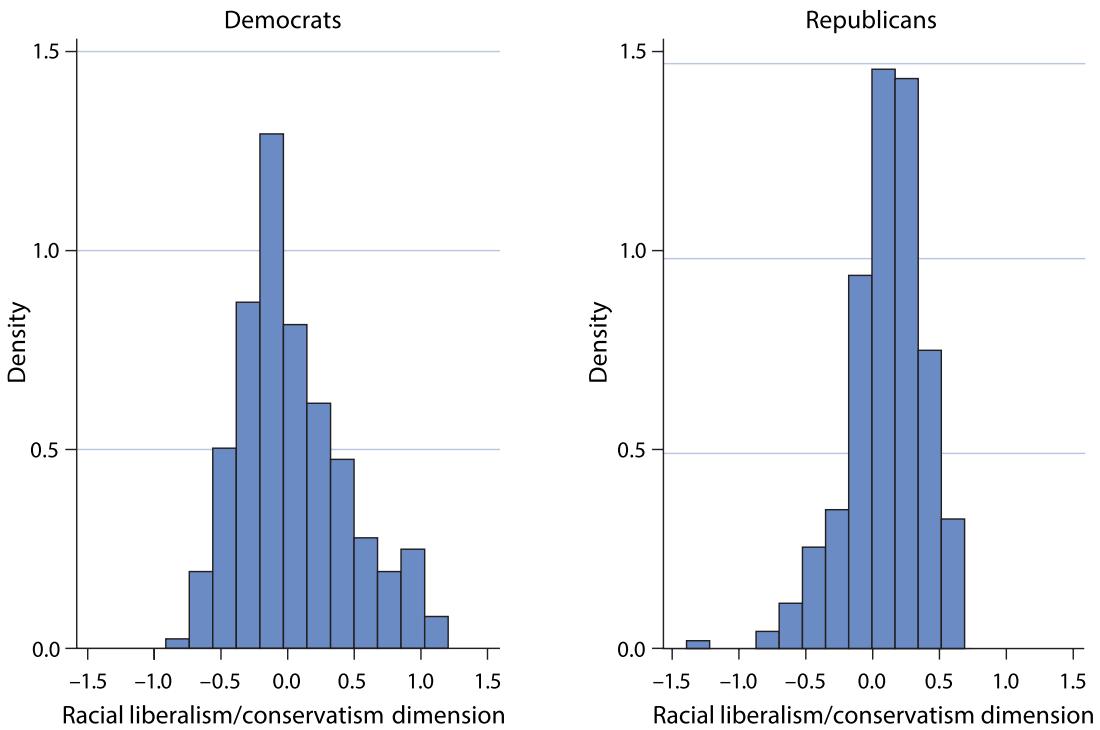
Finally, in some cases, we are interested in comparing the entire distributions of two variables rather than just the mean or median. One way to conduct such a comparison is to simply plot two histograms side-by-side. As an example, we compare the distribution of ideal points on the racial liberalism/conservatism dimension in the 112th Congress. When comparing across multiple plots, it is important to use the same scales for the horizontal and vertical axes in all plots to facilitate the comparison. For a cleaner plot, we aggregate the data into just 12 bins, using the `bin()` option.

```
. use congress, clear

. * party distributions on dimension 2
. histogram dwnom2 if party == "Democrat" & congress == 112, ///
>     xscale(range(-1.5(.5)1.5)) xlabel(-1.5(.5)1.5) ///
>     xtitle("Racial liberalism/conservatism dimension") ///
>     bin(12) title("Democrats") fcolor(none) color(black) name(d112, replace)
(bin=12, start=-.92500001, width=.17725)

. histogram dwnom2 if party == "Republican" & congress == 112, ///
>     xscale(range(-1.5(.5)1.5)) xlabel(-1.5(.5)1.5) ///
>     xtitle("Racial liberalism/conservatism dimension") ///
>     bin(12) title("Republicans") fcolor(none) color(black) name(r112, replace)
(bin=12, start=-1.381, width=.17125001)

. graph combine d112 r112
```



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 toward 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 scatterplot of quantiles where each point represents the same quantile. We can plot the median, upper quartile, and lower quartile of one sample against the corresponding quantiles of another sample. If two distributions are identical, then all quantiles have the same values and 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 suggest 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 are able to 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` command generates this plot by specifying the arguments `y` and `x`. First, though, we create separate `dwnom2` variables for Democrats and Republicans since they will be plotted at the same time.

```
. generate demdwnom2 = dwnom2 if party == "Democrat"
(6,420 missing values generated)
```

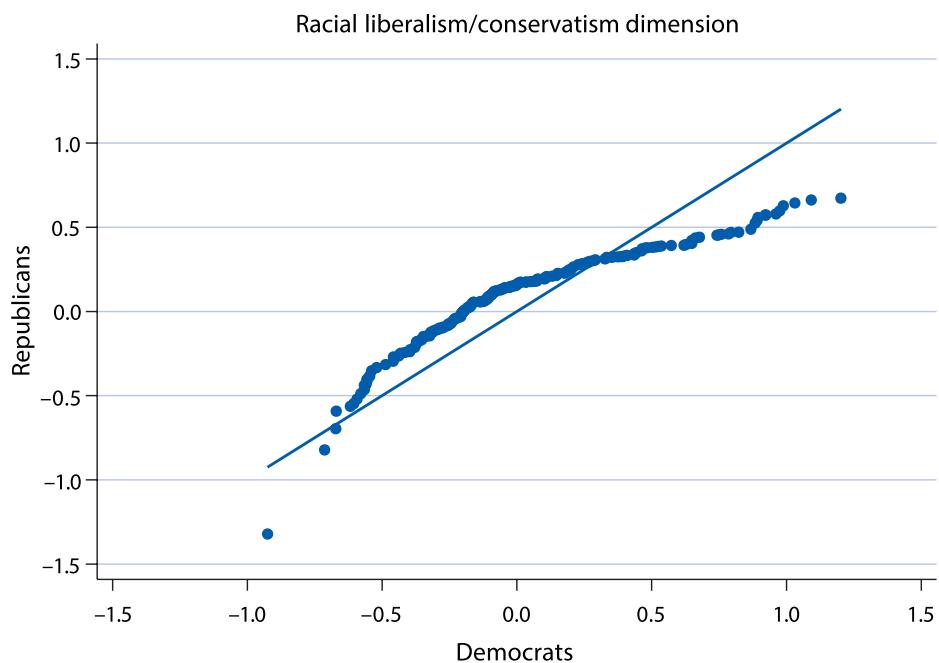
```

. generate repdwnom2 = dwnom2 if party == "Republican"
(8,151 missing values generated)

.

. qqplot repdwnom2 demdwnom2 if congress == 112, ///
> xlabel(-1.5(.5)1.5) ylabel(-1.5(.5)1.5) ///
> ytitle("Republicans") xtitle("Democrats") title("Racial liberalism/conservatism
dimension")

```



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 indicates 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 scatterplot 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 scatterplot 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*.

### 3.7.1 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 Euclidian 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 Steps 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 Stata, the initial locations of centroids are randomly selected.

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 Stata, we can standardize a variable using the `ege`n command with the `std()` function, which takes a single variable or expression.

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 two and four clusters, respectively. The command `cluster kmeans` implements the  $k$ -means algorithm in Stata.

The required argument of the `cluster kmeans` command is `k` (the number of clusters). Optional arguments include `name()` (the name of the cluster analysis results), `start()` (how to obtain the initial group centers), and `measure()` (the similarity or dissimilarity measure, with Euclidean distance serving as the default). Starting with the 80th Congress, we begin fitting the  $k$ -means algorithm with two clusters `k(2)` and use Stata's default start setting `krandom` that randomly selects  $k$  observations to serve as the starting centers for  $k$  groups. We set a seed of 12345 within the `krandom` specification. Setting a seed ensures that Stata will draw the same random numbers in its algorithm, which allows us to replicate the exact results that are based on randomization. The `cluster` command produces a new variable that assigns each observation into one of the two  $k$ -clusters. We save this as `kdwnom80`. We then run a second command to produce a similar variable (`kdwnom112`) for the 112th Congress.

```
. * k-means with 2 clusters, by Congress
. cluster kmeans dwnom1 dwnom2 if congress == 80, k(2) ///
>         start(krandom(12345)) name(kdwnom80)
. cluster kmeans dwnom1 dwnom2 if congress == 112, k(2) ///
>         start(krandom(12345)) name(kdwnom112)
```

We examine the final centroids of the resulting clusters using a two-cluster model. We can use the `tabstat` command to show the mean `dwnom1` and `dwnom2` scores for each cluster within each Congress. Each row in the results represents the axis for the corresponding center.

```
. * final centroids
. tabstat dwnom1 dwnom2, by(kdwnom80)

Summary statistics: mean
by categories of: kdwnom80

kdwnom80 |   dwnom1     dwnom2
-----|-----
1 | .1521266  -.343539
2 | -.056058   .7686304
-----|-----
Total | .0877108   .0005852

. tabstat dwnom1 dwnom2, by(kdwnom112)

Summary statistics: mean
by categories of: kdwnom112

kdwnom112 |   dwnom1     dwnom2
-----|-----
1 | .6753251   .0929671
```

2	-.39376	.029455
Total	.1926682	.0642935

We next compute the number 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
. tabulate party kdwnom80
```

party	kdwnom80		Total
	1	2	
Democrat	59	135	194
Other	2	0	2
Republican	247	3	250
Total	308	138	446

```
. tabulate party kdwnom112
```

party	kdwnom112		Total
	1	2	
Democrat	0	200	200
Republican	243	0	243
Total	243	200	443

We find that for the 112th Congress, the  $k$ -means algorithm with two clusters produces one cluster containing all Democrats 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 two 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 four clusters and visualize the results. We begin by fitting the four-cluster model to the 80th and 112th Congresses. Also, we can ask Stata to store the mean values for each cluster using the `keepcenters` option. This will add an observation for each cluster with the mean value of the variable in the cluster.

```
. * k-means with 4 clusters
. cluster kmeans dwnom1 dwnom2 if congress == 80, k(4) ///
> start(krandom(12345)) name(k4dwnom80)

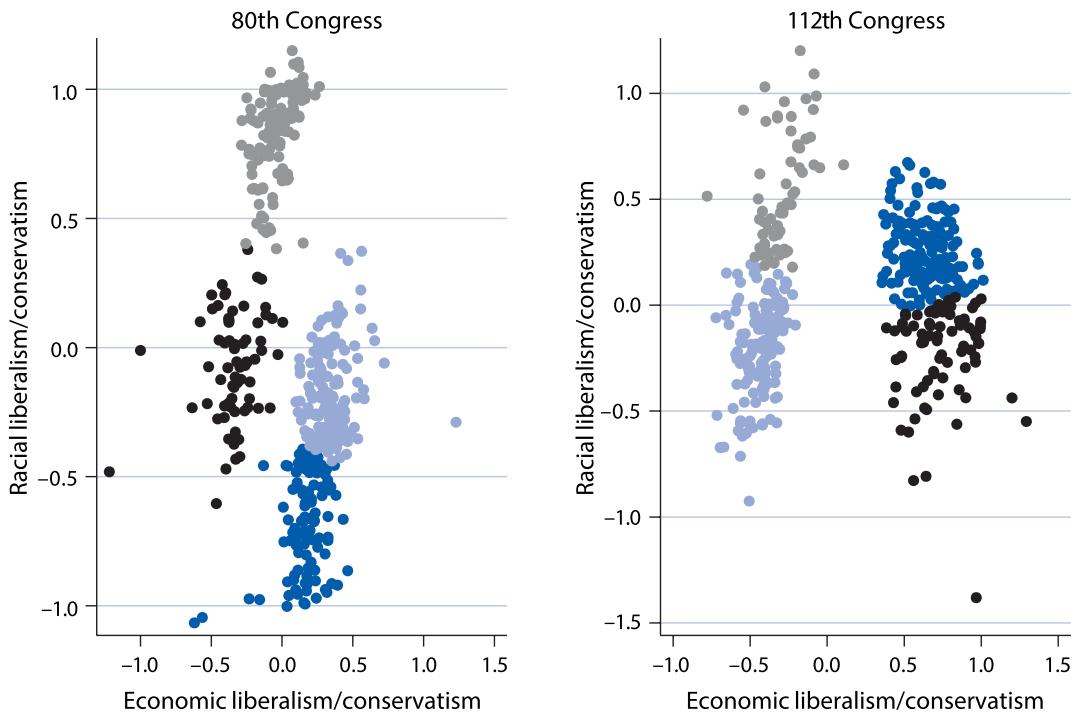
. cluster kmeans dwnom1 dwnom2 if congress == 112, k(4) ///
> start(krandom(12345)) name(k4dwnom112)
```

To visualize the results, we use the `twoway scatter` command to create a scatterplot. We create a plot for each of the clusters, separating each `scatter` command with `||`. When using a monochrome scheme like `s1mono`, this assigns a different shade and marker to the observations that belong to a different cluster. It will simply assign different colors in color graphs. See page C2 for the full-color version.

```

. * plotting results - 80th Congress
. scatter dwnom2 dwnom1 if k4dwnom80 == 1 || ///
>           scatter dwnom2 dwnom1 if k4dwnom80 == 2 || ///
>           scatter dwnom2 dwnom1 if k4dwnom80 == 3 || ///
>           scatter dwnom2 dwnom1 if k4dwnom80 == 4, ///
>           xtitle("Economic liberalism/conservatism") ///
>           ytitle("Racial liberalism/conservatism") ///
>           title("80th Congress") legend(off) name(cluster80, replace)
. * plotting results - 112th Congress
. scatter dwnom2 dwnom1 if k4dwnom112 == 1 || ///
>           scatter dwnom2 dwnom1 if k4dwnom112 == 2 || ///
>           scatter dwnom2 dwnom1 if k4dwnom112 == 3 || ///
>           scatter dwnom2 dwnom1 if k4dwnom112 == 4, ///
>           xtitle("Economic liberalism/conservatism") ///
>           ytitle("Racial liberalism/conservatism") ///
>           title("112th Congress") legend(off) name(cluster112, replace)
. graph combine cluster80 cluster112

```



The results show that the four-cluster model splits the Democrats into two clusters and the Republicans into two clusters. Within each party, the division between the two 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 or 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 to provide socially acceptable responses.

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. 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 **scatterplot** plots one variable against the other. Finally, unlike

scatterplots, **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.<sup>4</sup> 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 data files: a reshaped version of the original data set in which every observation corresponds to a unique respondent, `gayreshaped.dta` (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.dta` (see table 3.6). Note that the feeling thermometer measures how warmly respondents feel toward 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? 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 `pwcorr` command performs a pairwise correlation. This means that the `pwcorr` command 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 scatterplot 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

<sup>4</sup>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 toward gay couples in wave 1 (0–100)
therm2	survey thermometer rating of feeling toward gay couples in wave 2 (0–100)
therm3	survey thermometer rating of feeling toward gay couples in wave 3 (0–100)
therm4	survey thermometer rating of feeling toward gay couples in wave 4 (0–100)

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 toward gay couples (0–100)

completely different respondents. We use the data contained in `ccap2012.dta` and `gayreshaped.dta` 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. What proportion of the sample in each study have a feeling thermometer rating of 50, the neutral point? What happens when we recode missing values to this neutral point? Compare the respective histograms. To facilitate the comparison across histograms, use the `bin()` argument in the `histogram` command to keep the bin sizes equal cross histograms. Briefly comment on the results.

5. A more direct way to compare the distributions of two samples is through a *quantile-quantile plot*, or *Q-Q 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.<sup>5</sup> 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.

<sup>5</sup>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

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.

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.dta`, 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, label the values of `self` accordingly (e.g., 4 is “A lot of say”) and 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) that 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. Does the severity of the measurement problem we have identified depend on the age of respondents? Is it 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.

**Table 3.8.** United Nations Ideal Point 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

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.<sup>6</sup> Table 3.8 presents the names and descriptions of the variables in the data set contained in the Stata file `unvoting.dta`.

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' 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 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 `centile` command 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 `collapse` command 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-United States.

<sup>6</sup>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.

What are the most pro-Russia 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 United States' and Russia's 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 `inlist()` function introduced in section 1.3.2 is useful here. It can be combined with the `generate` or `replace` commands. This function returns a value of 1 if the value of the variable first specified in the parentheses equals any of the values listed thereafter. A maximum of 255 numeric values can be included with the `inlist()` option and a maximum of 10 string values. Consequently, you will need more than one `inlist()` command to include all former Soviet countries. 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 toward 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 unique *Electoral College* system of the United States 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 the Republican

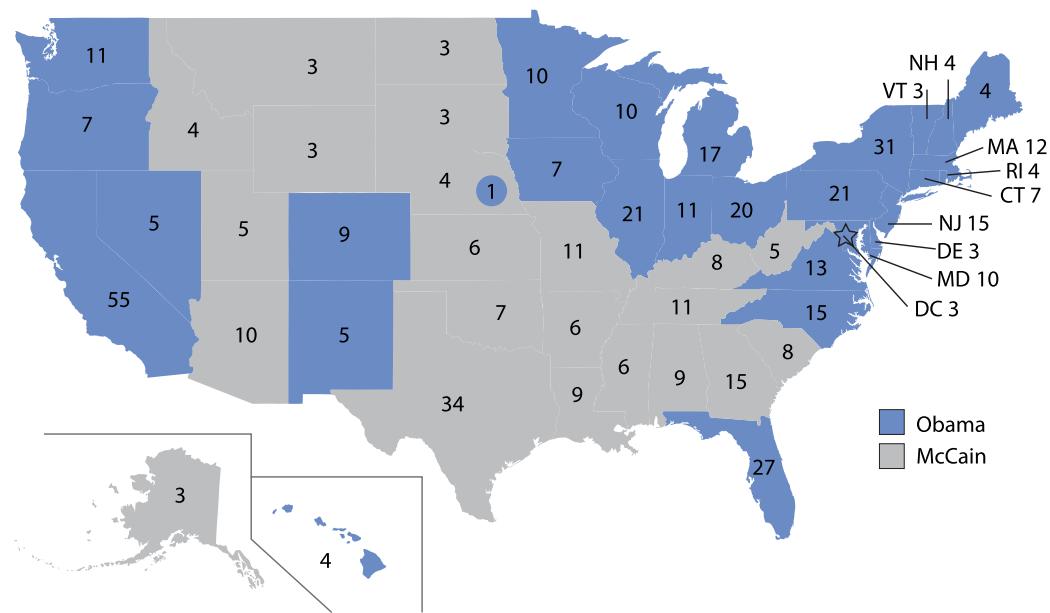


Figure 4.1. Electoral College Map of the 2008 US Presidential Election. Source: Wikipedia

candidate John McCain received 173 votes (red states).<sup>1</sup> 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. In this chapter, 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 that help us code more efficiently in Stata: macros and loops.

#### 4.1.1 MACROS

Macros and loops are two important tools in Stata that simplify the coding process when repetition is involved. Both can save us significant time and effort, especially when used together. Like *scalars* introduced in section 2.5.1, *macros* store values that can be referred to at later points in the code, allowing us to avoid tedious retyping. While scalars store single values, macros can store multiple values that are read as one long string. The macro then becomes shorthand for that string, providing a representation any time it is called upon in the code. Scalars are more accurate when storing single values such as long numbers because the storage is direct. There is no translation into an abbreviated representation. However, macros are more versatile and can handle far more complex tasks, including those that require

<sup>1</sup>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.

repetition. They can represent either numeric values or character values, depending on our purpose.

Stata offers `local` and `global` macros. Local macros are temporary. When they are created in a do-file, they are dropped from memory once the file has finished running. Global macros remain in memory, even after a do-file executes. Both types store string values under an assigned macro name. In the following syntax, we create a local macro called `book`, using the `local` command. We specify the macro name immediately following the `local` command. The string value that we want to store must be surrounded by double quotations. We then call upon the content of the local macro by placing the name between a preceding grave accent (or backtick) and followed by an apostrophe (or single quote).

```
. local book "Quantitative Social Science: An Introduction with Stata"
. display "I am currently reading `book'."
I am currently reading Quantitative Social Science: An Introduction with Stata.
```

Following the same general syntax, we use the `global` command to create a global macro (`global macroname "string value"`). However, when we call upon a global macro within our code, we place a dollar sign before the macro name.

```
. global book2 "Quantitative Social Science: An Introduction with Stata"
. display "I am currently reading $book2."
I am currently reading Quantitative Social Science: An Introduction with Stata.
```

Both `local` and `global` macros can also store expressions, where we use the equals sign (=) instead of double quotations in the assignment. Thinking back to the arithmetic operators reviewed in chapter 1, we could make calculations with raw numbers, as shown in the example below, or include numeric variables as our code becomes more complex.

```
. local math = 5 + 10 / 2
. display `math'
10
```

Because they remain in memory, global macros are more likely to generate errors from naming or other conflicts. However, the memory-based distinction between local and global macros does not apply if the user is typing all commands directly into the command console. In that case, both local and global macros will remain in memory until the end of the session. But, as reviewed in section 1.3.8, using the command console alone is not advised. It is generally good practice to use local macros unless there is a necessary reason to use global macros. We can see a list of existing macros in memory by running the command `macro dir`. Macros can be cleared by using `macro drop macroname`.

Local and global macros typically make it easier to update our code when alterations are required. Changing one line of code in our do-file (where the macro is defined) will consistently update all pieces of code where the macro is used. This not only alleviates the need for retying but it also leaves us less prone to errors.

Before continuing, it should be noted that Stata offers a number of other macros. Later in this chapter, we will use the  `tempfile` command, which creates a temporary data file that is erased after the execution of a do-file. To see more about macros, type `help extended_fcn`.

#### 4.1.2 LOOPS

In many situations, we want to repeat the same operations multiple times where only small changes occur to the operations each time. For example, we may want to change the variable, subgroup, or year included in our calculation. 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 must be performed a number of times. To avoid retying nearly identical blocks of code over and over again for each state, we can use a *loop*. A *loop* is a programming construct that allows us to repeatedly execute similar code blocks in a compact manner. Stata has several different syntaxes for loops, depending on whether we want to loop over numbers, variables, or strings. The two key commands for loops in Stata are `forvalues` and `foreach`.

The `foreach` command allows us to loop over strings, macros (local or global), existing or potential variable names, and numbers. This option is appropriate for our example where we want to repeat calculations for multiple states. The general structure of a `foreach` loop is `foreach element of [list type] list name or contents`. There are five list types: local, global, varlist, newlist, and numlist. List types local and global specify already defined macro names, varlist contains a list of existing variable names, newlist a list of new (non-existing) variables, and numlist is a list of numbers. The following syntax shows the pseudocode using `foreach` with the `varlist` suboption.

```
foreach v of varlist variable1 variable2 variable3 {
    command1
    command2
    ...
    commandN
}
```

Here, the collection of commands from `command1` through `commandN` is repeated for each element `v` of the variables that follow `varlist`. During each of these *iterations*, `v` takes on the corresponding value from the list, starting with the first element (`variable1`) and then ending with the last (`variable3`). Braces `{` and `}` are used to denote the beginning and end of the body of the loop.

Let's try some examples with the other potential list types. Remember that if we use local or global macros, these must be defined before running the loop, or the loop will not perform any calculations. In addition, within a loop, strings of characters (e.g., words) stored in a local or global macro will be read as individual elements when separated by a space.

```

. local mylocal "string1 string2 string3" // must define local if not previously
   defined

. foreach v of local mylocal {
2.         display "This list contains string `v`."
3. }

This list contains string string1.
This list contains string string2.
This list contains string string3.

.

. foreach v of numlist 1 / 4 {
2.         display "This list contains number `v`."
3. }

This list contains number 1.
This list contains number 2.
This list contains number 3.
This list contains number 4.

.

. foreach v of newlist newvar1 newvar2 newvar3 {
2.         display "Here we can create new variable `v`."
3. }

Here we can create new variable newvar1.
Here we can create new variable newvar2.
Here we can create new variable newvar3.

```

We can also use `forvalues i = number range` to loop over a range of numeric values. In fact, `forvalues` is strictly limited to numbers or sets of numbers, unlike `foreach`, which also handles strings. There are two main ways to specify the steps in the range. A slash (/) will tell Stata to increment the loop by a value of 1. We could also use parentheses to increment the loop by values other than 1. For example, to loop over the range 0 to 100 in increments of 10, we would write `forvalues i = 0(10)100`. The syntax for the `forvalues` loop is similar to that of a `foreach` loop.

```

. forvalues i = 1 / 4 {
2.         generate newvar`i' = 2 + `i'
3.         display newvar`i'
4. }

3
4
5
6

```

**Table 4.1.** 2008 US Presidential Election Data.

Variable	Description
state	abbreviated name of the state
statename	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
obamapoll	predicted support for Obama (percentage)
mccainpoll	predicted support for McCain (percentage)
pollster	name of the organization conducting poll
middate	middate of the period when the poll was conducted

In general, `forvalues` loops are faster when looping through numeric values. Similarly, it is more efficient to store variable names in a local macro, rather than using the `varlist` option when running `foreach` loops. Not only will this use less memory, but also a local macro is easier to update if the variable list will be referenced later in the code.

Loops can be nested within other loops, and their versatility means that they can perform tasks across variables, observations, or even files. Commands within a loop can be annotated as with any other code in Stata. When you start a loop (or related functions) in Stata's do-file editor, a good habit is to indent the contents of the loop (i.e., the commands), keeping the closing brace lined up with the corresponding `for` function that began the loop. This makes the code easier to read and debug (i.e., identify and remove errors from one's code).

#### 4.1.3 POLL PREDICTIONS

We now 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 Stata data file `pres08.dta` contains the election results by state. In addition, we have the file `polls08.dta`, which contains many polls within each state leading up to the election.<sup>2</sup> The names and descriptions of the variables in these data sets are given in tables 4.1 and 4.2, respectively. We begin by merging the two data sets by state. Within each data set, we then create a variable that represents Obama's vote margin over McCain in percentage points, naming these `presmargin` and `pollmargin`.

<sup>2</sup>The polling data were obtained from <http://electoral-vote.com>.

```
. cd prediction
. use pres08, clear
```

```
. merge 1:m state using polls08
Result # of obs.
-----
not matched 0
matched 1,332 (_merge==3)
```

```
. generate presmargin = obama - mccain
. generate pollmargin = obamapoll - mccainpoll
```

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 identify the day closest to Election Day within each state, we first create a new variable using the `date()` function. The `date()` function allows us to convert the string value of our current date variable (`middate`) into a numeric variable we can use to calculate the number of days between a poll and Election Day. Within the `date()` function, we must correctly specify the order in which the day, month, and year are presented in the string value. In our example, `middate` is formatted as year-month-day ("YMD"). In other cases, the date format may be month-day-year ("MDY") or day-month-year ("DMY"). Stata converts the string into an integer that counts the number of days between the observed date and January 1, 1960. If a poll was conducted on January 10, 1960, for example, the new date variable will be assigned a value of 9 (i.e., 9 days between January 1, 1960 and January 10, 1960). Dates preceding January 1, 1960 will have a negative integer value. Stata offers functions for the conversion of other units of time, including seconds, weeks, months, and years. Details on additional options can be found by typing `help datetime`.

Returning to our example, we store the converted numeric value in a new date variable called `polldate`. We then use the `format` command with the `%td` specification to tell Stata that our variable measures days. This will also transform the display of the integer values into a readable date format (e.g., 17793 becomes 18sep2008).

```
. * create a date variable
. generate polldate = date(middate, "YMD")
. format polldate %td
```

To calculate the number of days to the election, we create the variable `daystoelection`. We compute this as the difference in days between the polling date (`polldate`) and Election Day (November 4, 2008). We could have created a new variable to store the date of the election,

but here we include the `date()` function in our calculation to illustrate how some functions can be directly included and combined with arithmetic operations in the generation of new variables. Before we subset the data, we save our recoded data set as a new file to use later in this section.

```
. * compute the number of days to Election Day
. generate daystoelection = date("2008-11-04", "YMD") - polldate
. * save recoded data as new file
. save poll_pres08, replace
file poll_pres08.dta saved
```

We can now compute the mean of poll predictions that occurred on the day closest to the election in each state. We identify the day closest to the election by using the `min()` function with the `egen` command. Adding `bysort state:` to the start of our command ensures that the minimum value of `daystoelection` is calculated separately for each state, rather than identifying the minimum value for the sample as a whole. We subset the data to only the most recent polls (i.e., where `daystoelection` equals the minimum value) using the `keep` command. The `collapse` command then computes summary statistics for the entire sample by state. The mean number of Electoral College votes and vote shares remain the same within states, but when more than one poll was held on the day most recent to the election, `collapse` provides the average. For clarity, we rename the mean polling margin `pollpred` by specifying `pollpred = pollmargin` in our command.

```
. * calculate and subset to most recent poll
. bysort state: egen mindays = min(daystoelection)
. keep if daystoelection == mindays
(1,277 observations deleted)

.
. * calculate and store mean of poll predictions
. collapse (mean) ev presmargin pollpred = pollmargin, by(state)
```

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*.

```
. * prediction error of latest polls in each state
. generate errors = presmargin - pollpred
.

. * mean prediction error
```

```
. summarize errors
```

Variable	Obs	Mean	Std. Dev.	Min	Max
errors	51	1.062092	5.870544	-16	16

The result shows that, on average and 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.

```
. * square the errors
. generate errors_sq = errors^2
. * calculate root of mean squared errors
. summarize errors_sq, meanonly
. display sqrt(r(mean))
5.9089405
```

We obtain the mean squared error by using the `summarize` command, which stores the resulting average in `r(mean)`. The `meanonly` option suppresses the output display. The result indicates that the average magnitude of each poll prediction error is about 6 percentage points.

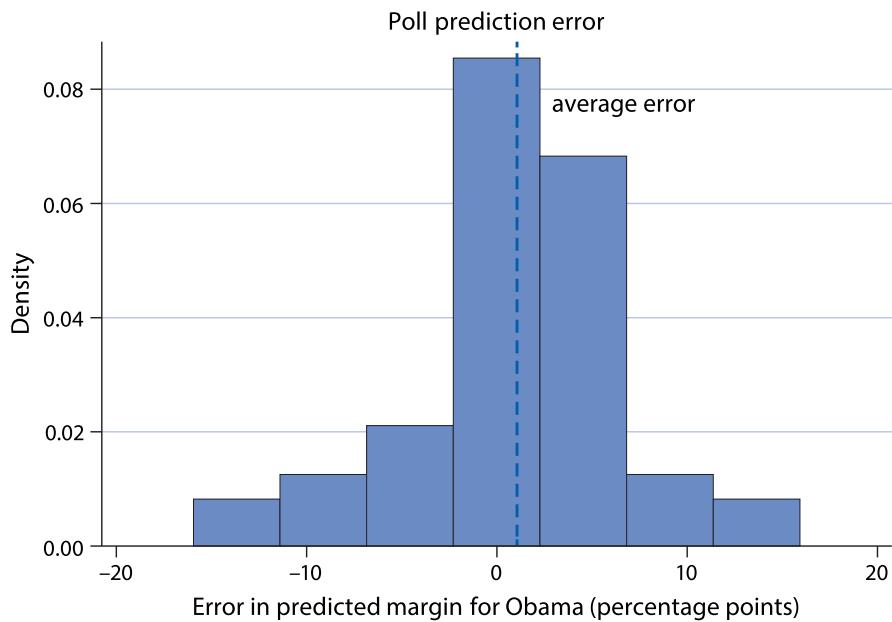
The **prediction error** is defined as

$$\text{prediction error} = \text{actual outcome} - \text{predicted outcome}$$

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 `histogram` command (see section 3.3.2). We use `xline` to plot the average error, stored in `r(mean)`, as a vertical line.

```
. * histogram
. summarize errors, meanonly
. histogram errors, xline(`r(mean)', lpattern(dash)) ///
>           xtitle("Error in predicted margin for Obama (percentage points)") ///
>           title("Poll prediction error") ///
>           text(.08 6.5 "average error", color(blue)) ///
>           fcolor(none) color(black)
(bin=7, start=-16, width=4.5714286)
```



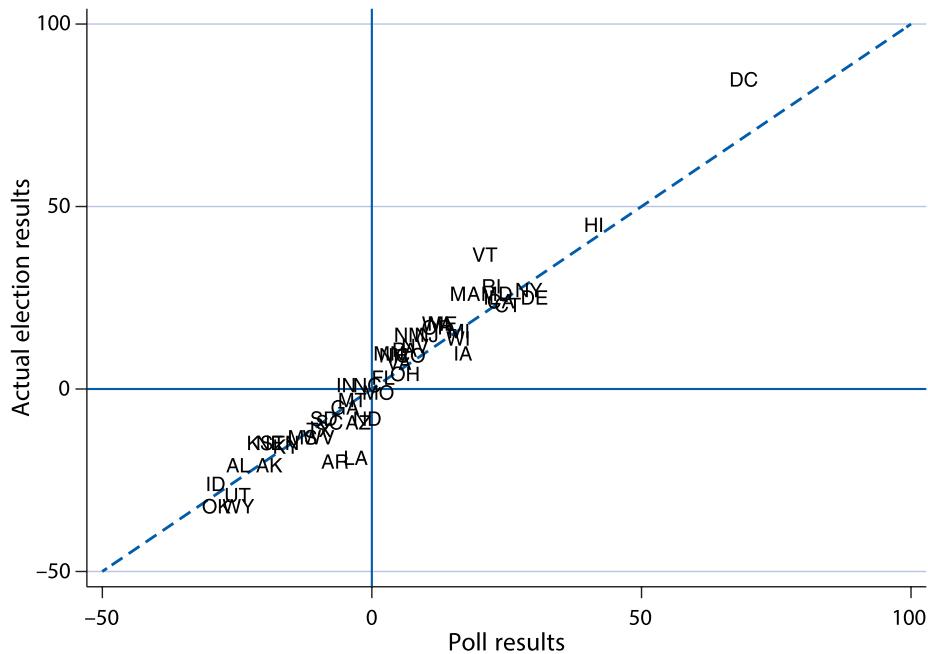
The histogram shows that the poll prediction error varies widely from one state to another. However, most errors are relatively small and larger errors are less likely to occur, yielding a *bell shaped* distribution around zero.

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`. States with accurate polling predictions of the election outcome would fall on the 45-degree line. The states above (below) the 45-degree line indicate that the poll predictions were too favorable toward Obama (McCain).

To add text labels to the markers in our graph, we include the `mlabel()` option with the variable name in parentheses (here, `state`). This tells Stata to add the label next to the marker symbol. We can control the placement of the label with the `mlabposition()` option. Because we want to replace the symbol with the text, we place the state label in the middle. The `mlabposition()` placements are based on a clock. The top-middle is "12" and the bottom-middle is "6". To the left of the symbol is "9" and to the right is "3". We place the text in the very middle, which is referred to by "0". The `msymbol(i)` option makes the marker symbol invisible so that it does not overlap with the text label.

We then add a 45-degree line using the `scatteri` command. Rather than plotting a graph based on variables, `scatteri` lets us place our own values in the command in the form of “y-coordinate x-coordinate.” We add in “-50 -50” as the starting point and “100 100” as the end. Additional graphs must be separated by double bars ||. The `connect(1)` option adds a straight line between the two points while `lpattern(dash)` makes the line dashed. It can be useful to change or remove some of these specifications to better see what each component changes in the graph. In the current example, the x-coordinates and y-coordinates represent the poll predictions and the actual Obama margins.

```
. scatter presmargin pollpred, xline(0) yline(0) ///
>          xtitle("Poll results") ytitle("Actual election results") ///
>          mlabel(state) mlabposition(0) msymbol(i) legend(off) || ///
>          scatteri -50 -50 100 100, connect(1) lpattern(dash)
```



Although the poll prediction is grossly inaccurate for states like the District of Columbia (DC) and Vermont (VT), 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 `pollpred` and `presmargin` for each state. The function returns 1 if positive (Obama wins) and -1 if negative (McCain wins). It returns 0 in the case of a tie.

```
. * which state polls called wrong?
. tabulate state if sign(pollpred) != sign(presmargin)

      state |   Freq.    Percent     Cum.
              |-----|
              IN |       1      33.33    33.33
              MO |       1      33.33    66.67
              NC |       1      33.33   100.00
              |
              Total |      3     100.00

.

. * what was the actual margin for these states?
. list state presmargin if sign(pollpred) != sign(presmargin)
```

	state	presma_n
16.	IN	1
25.	MO	-1
28.	NC	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%.

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 Indiana (IN) and North Carolina (NC) are false negatives. Table 4.3 presents a *confusion matrix* where various 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

**Table 4.3.** Confusion Matrix.

		Actual outcome	
		Positive	Negative
		Positive	
Predicted outcome	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

There are two types of correct classification, true positive and true negative. Similarly, false positive and false negative are two kinds of misclassification.

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.<sup>3</sup> Using `r(sum)`, we can add all total electoral votes across observations, or states, meeting our criteria (i.e., a positive margin predicting an Obama victory).

- . \* actual results: total number of electoral votes won by Obama
- . summarize ev if presmargin > 0

Variable	Obs	Mean	Std. Dev.	Min	Max
ev	29	12.55172	11.00515	3	55

- . display `r(sum)`

364

- . \* poll prediction
- . summarize ev if pollpred > 0

Variable	Obs	Mean	Std. Dev.	Min	Max
ev	28	12.46429	11.19683	3	55

- . display `r(sum)`

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 Stata file `pollsUS08.dta`. 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. 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. We begin by loading the data set and creating a date variable.

<sup>3</sup>As noted earlier, Obama received one vote from Nebraska even though he lost the statewide vote.

```
. * load the data
. use pollsUS08, clear
. * create date variable
. generate polldate = date(middate, "YMD")
. format polldate %td
```

We can calculate the average of polls occurring within the last 7 days by using *time-series operators* in Stata. As introduced in chapter 1, we refer to values in rows preceding or following any individual observation as lags and forwards, respectively. In time-series data, the rows correspond to different values of time (in our case, days). We previously used the built-in variable `_n` to access lagged values. However, we can also simply use the `L.` and `F.` prefixes before any variable to obtain the lagged and forward values and utilize additional features in Stata.

To use the time-series operators and commands in Stata, it is important that values of our time variable are not repeated—or, in the case of *panel data*, the same value of time (e.g., year) can reoccur, but not within the main unit of analysis (e.g., states or individuals). Because we are not using panel data and multiple polls were conducted on some days, we calculate the total number of polls conducted each day using `egen` and the `count()` function. For each day, we then add all polling numbers for Obama and McCain, separately, using the `collapse` command with the `sum()` function. We also include the total number of polls per day, `polls`, in our `collapse` command to use later in our calculation of the seven-day averages.

```
. * calculate how many polls per day
. bysort polldate: egen polls = count(polldate)
. * create summary data set
. collapse polls (sum) mccainpoll obamapoll, by(polldate)
```

Now that we have only one observation for each day, we can tell Stata that we are using *time-series* data by setting the `tsset` command. We must specify the time variable by which the data are ordered (`polldate` in our example). In the case of panel data, we would also specify the identifier variable that defines our unit of analysis. The `tsset` declaration will look for consistent time gaps between observations. Within our polling data, there are some days on which no polls were held. Consequently, some days are not included and do not have a row. We fill these gaps using the `tsfill` command.

```
. * declare time-series data and fill gaps
. tsset polldate
    time variable: polldate, 04jul2007 to 03nov2008, but with gaps
              delta: 1 day
. tsfill
```

Once the data have been declared as time-series, we can not only refer to lags and forwards by using simple prefixes, but there are a number of other available commands tailored for use with time-series data. We demonstrate with the `tsegen` command, which extends `egen` functions to time-series data. The `tsegen` module can be installed by typing `ssc install tsegen` into the command line. We use the `rowtotal()` function with the syntax `L(0/6).variable` to sum up all values from all polls conducted in the last seven days, by candidate. `L0` refers to the current day and `L6` is the sixth lag (i.e., six days prior).

```
. * sum total polling numbers and polls occurring in past 7 days using lags
. tsegen obama_7day = rowtotal(L(0/6).obamapoll)
. tsegen mccain_7day = rowtotal(L(0/6).mccainpoll)
. tsegen polls_7day = rowtotal(L(0/6).polls)
```

To obtain an average, the seven-day sum of polls for Obama and McCain must be divided by the total number of polls conducted that week. Before plotting the 90-day results, we review another way to calculate the seven-day averages.

```
. * divide 7-day polling sums by 7-day total number of polls
. replace obama_7day = obama_7day / polls_7day
(436 real changes made, 55 to missing)

. replace mccain_7day = mccain_7day / polls_7day
(436 real changes made, 55 to missing)
```

An alternative way to compute the seven-day average is through a loop. In the syntax that follows, we create a loop that goes through 90 iterations. Within each iteration, the data will be subsetted (using `if`) to only polls conducted within the past seven days, including the day itself. We then use `summarize` to obtain the sum of all polls for that seven-day period, which we divide by the total number of polls conducted over that time. As an 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), the `summarize` command will include only observations where `daystoelection` takes a value of less than or equal to 7 ( $=90 - 90 + 7$ ) and greater than 0 ( $=90 - 90$ ). We nest this 7-day calculation inside another loop that performs the calculations separately for each candidate with the use of a local macro called `pres`.

```
. generate daystoelection = date("2008-11-04", "YMD") - polldate
. generate obama2_7day = 0
. generate mccain2_7day = 0
```

```

. generate polls2_7day = 0
. local pres "obama mccain"
. foreach cand of local pres {
    2.         quietly forvalues i = 1 / 90 {
    3.             summarize polls if daystoelection <= (90 - `i' + 7) & ///
    >                         daystoelection > 90 - `i'
    4.             summarize `cand'poll if daystoelection <= (90 - `i'+ 7) & ///
    >                         daystoelection > 90 - `i'
    5.             replace `cand'2_7day = r(sum) / polls_7day if daystoelection
    == 91 - `i'
    6.         }
    7. }

```

The means of the seven-day averages obtained from the loop equal the means calculated with the time-series approach.

```

. * comparing the means between the time-series and loop approach
. summarize obama_7day mccain_7day ///
>           obama2_7day mccain2_7day if daystoelection <= 90

```

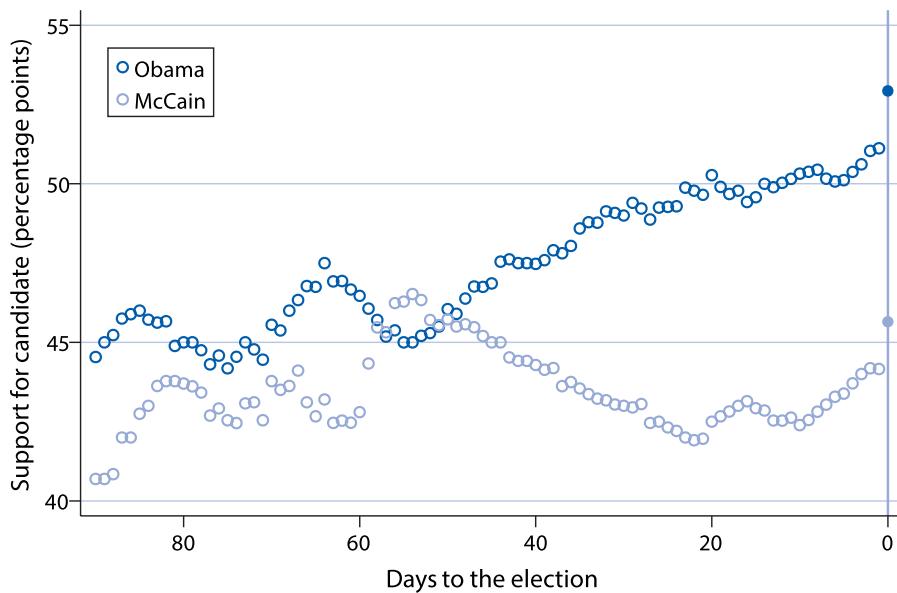
Variable	Obs	Mean	Std. Dev.	Min	Max
obama_7day	90	47.43964	2.073134	44.18182	51.12
mccain_7day	90	43.4655	1.250082	40.69231	46.52174
obama2_7day	90	47.43964	2.073134	44.18182	51.12
mccain2_7day	90	43.4655	1.250082	40.69231	46.52174

The time-series operators and loop will produce the same *time-series plot*. This can be verified by substituting the loop's `obama2_7day` and `mccain2_7day` variables with the time-series variables (`obama_7day` and `mccain_7day`) in our `scatter` command. Because we want to display the results according to days away from the election, we use the `daystoelection` variable, not the `polldate` variable. We provide a countdown on the horizontal axis using the `xscale(reverse)` option, which places 90 days prior to the election day as the leftmost value and the election day as the rightmost value.

```

. * create scatterplot using days to election variable
. scatter obama2_7day mccain2_7day daystoelection if daystoelection <= 90, ///
>           xlabel(0(10)90) msymbol(Oh Oh) mcolor(blue red) || ///
>           scatteri 52.93 0, mcolor(blue) || scatteri 45.65 0, mcolor(red) ||
>           xlabel(0(20)80) xscale(reverse) xtitle("Days to the election") ||
>           ytitle("Support for candidate (percentage points)") ||
>           legend(label(1 "Obama") label(2 "McCain") order(1 2))

```



The figure (see also page C3) demonstrates the reasonable accuracy of preelection polls in terms of margin. Indeed, the Election Day margin (the difference between the 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 over the course of the 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.<sup>4</sup> 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.

<sup>4</sup>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.

Which person is the more 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.

**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
wparty	party of the winner
lparty	party of the loser
dvotes	number of votes for the Democratic candidate
rvotes	number of votes for the Republican candidate
dcomp	competence measure for the Democratic candidate
rcomp	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 file `face.dta` 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 facial expression alone.

We begin our analysis of the facial appearance experiment data by creating a *scatterplot* of the competence measure against election outcomes. To do this, we first 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
. use face, clear

. * two-party vote share for Democrats and Republicans
. generate dshare = dvotes / (dvotes + rvotes)
. generate rshare = rvotes / (dvotes + rvotes)
. generate diffshare = dshare - rshare

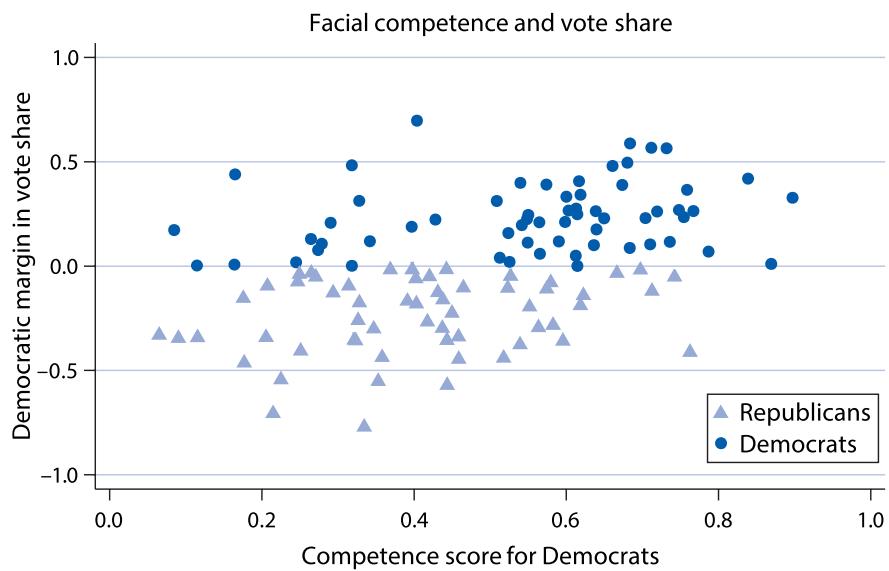
```

Next, we use the `scatter` command to generate a scatterplot. To make the symbols more informative, we can change them based on variables in our data set. We specify the color of points to be plotted using the `mcolor` option and separately plot races with Republican winners (in red) and races with Democratic winners (in blue) using `||`. The full-color version can be found on page C3. The plot shows a mild upward trend in the Democratic margin as the competence score for Democrats increases.

```

. scatter diffshare dcomp if wparty == "R", mcolor(red) || ///
>           scatter diffshare dcomp if wparty == "D", mcolor(blue) ///
>           xtitle("Competence score for Democrats") ///
>           ytitle("Democratic margin in vote share") ///
>           title("Facial competence and vote share") ///
>           legend(label(1 "Republicans") label(2 "Democrats"))

```



#### 4.2.2 CORRELATION AND SCATTERPLOTS

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 likely to be above (below) its mean when the other variable is above its own mean.

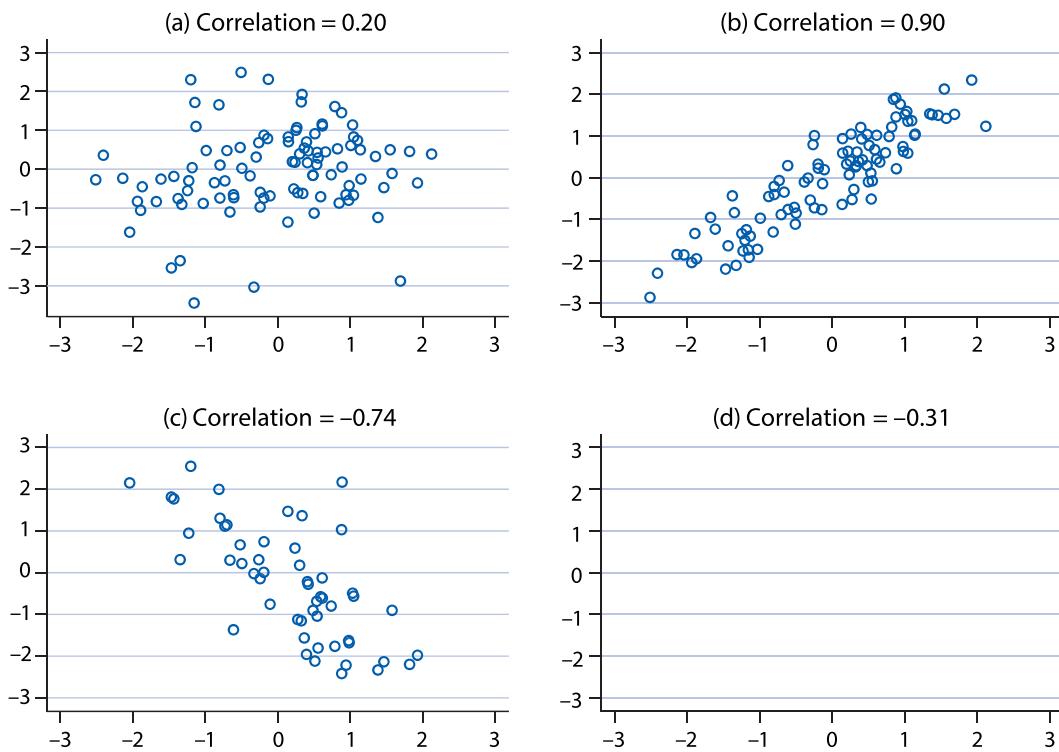


Figure 4.3. Correlation Coefficients and Patterns of the Data Cloud in Scatterplots.

The upward-sloping data cloud in the facial appearance scatterplot suggests a positive *correlation* between perceived competence and vote share differential. The `correlate` command computes the correlation coefficient.

```
. correlate dcomp diffshare
(obs=119)

      dcomp      1.0000
diffshare    0.4328    1.0000
```

This correlation of about .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 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 upward (downward) 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 figure 4.3d, 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 upward trend in the data cloud in a scatterplot implies a positive correlation, whereas a downward 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 reviewed, correlation describes a linear relationship between two variables. However, such a relationship is best characterized using the following *linear model*:

$$Y = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} X + \underbrace{\epsilon}_{\text{error term}}. \quad (4.1)$$

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. Any line can be defined by the *intercept*  $\alpha$  and the *slope* parameter  $\beta$ . The intercept  $\alpha$  represents the average value of  $Y$  when  $X$  is zero. The slope  $\beta$  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,  $\epsilon$ , 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  $\alpha$  and  $\beta$  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  $\alpha$  and  $\beta$ , respectively. Once we obtain the estimated values of coefficients  $\alpha$  and  $\beta$ , 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  $\hat{Y}$ , using the regression function

$$\hat{Y} = \hat{\alpha} + \hat{\beta}x. \quad (4.2)$$

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

$$\epsilon = Y - \hat{Y}. \quad (4.3)$$

Notice that the residual is represented by  $\epsilon$  with a hat. Since the error term  $\epsilon$  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,  $\epsilon$  is the error (or disturbance) term, and  $(\alpha, \beta)$  are the coefficients. The slope parameter  $\beta$  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  $\hat{Y} = \hat{\alpha} + \hat{\beta}x$ . The difference between the observed outcome and this fitted or predicted value  $\hat{Y}$  is called the residual and is denoted by  $\hat{\epsilon} = Y - \hat{Y}$ .

To fit a linear regression model in Stata, we use the `regress` command. This command takes outcome variable  $Y$ , and predictor variable  $X$ . 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.

.	*	get estimated coefficients				
.		regress diffshare dcomp				
Source		SS	df	MS	Number of obs	= 119
Model		1.91478729	1	1.91478729	F(1, 117)	= 26.96
Residual		8.30866377	117	.07101422	Prob > F	= 0.0000
					R-squared	= 0.1873
Total		10.2234511	118	.086639416	Adj R-squared	= 0.1803
					Root MSE	= .26648
diffshare		Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
dcomp		.6603814	.1271766	5.19	0.000	.4085148 .9122481
_cons		-.3122259	.0659623	-4.73	0.000	-.4428607 -.181591

The output shows that the estimated intercept is  $-.3122$  whereas the estimated slope is  $.6604$ . 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  $6.6 (= .6604 \times 10)$  percentage points.

Adding `coeflegend` to the end of our regression command shows us how we can refer to the stored results. For example, we can obtain the stored estimated coefficients ( $\hat{\alpha}, \hat{\beta}$ ) using `_b[varname]` and the stored intercept using `_cons` or `_b[_cons]`.

```
. * display estimated coefficients and intercept
. regress diffshare dcomp, coeflegend

Source          SS           df           MS      Number of obs   =    119
                F(1, 117)     =    26.96
Model          1.91478729      1  1.91478729  Prob > F     =  0.0000
Residual       8.30866377     117  .07101422  R-squared      =  0.1873
Total          10.2234511    118  .086639416  Adj R-squared  =  0.1803
                                         Root MSE     =  .26648

diffshare      Coef.  Legend
dcomp          .6603814  _b[dcomp]
_cons         -.3122259  _b[_cons]

. display _b[_cons]
-.31222587

. display _b[dcomp]
.66038144
```

To store the predicted or fitted values  $\hat{Y}$  in a new variable, we use the `predict` command with the `xb` option, which tells Stata that we would like the predicted values. The `xb` abbreviation can be thought of as a rough translation for “predictor X with coefficient *beta*.” To obtain the residuals, we would replace `xb` with `residuals`.

```
. * get fitted or predicted values
. predict fitted, xb
. list fitted in 1 / 6

fitted
1.  .0606041
2.  -.0864334
3.  .0921706
4.  .0453924
5.  .1369869
6.  -.1005721
```

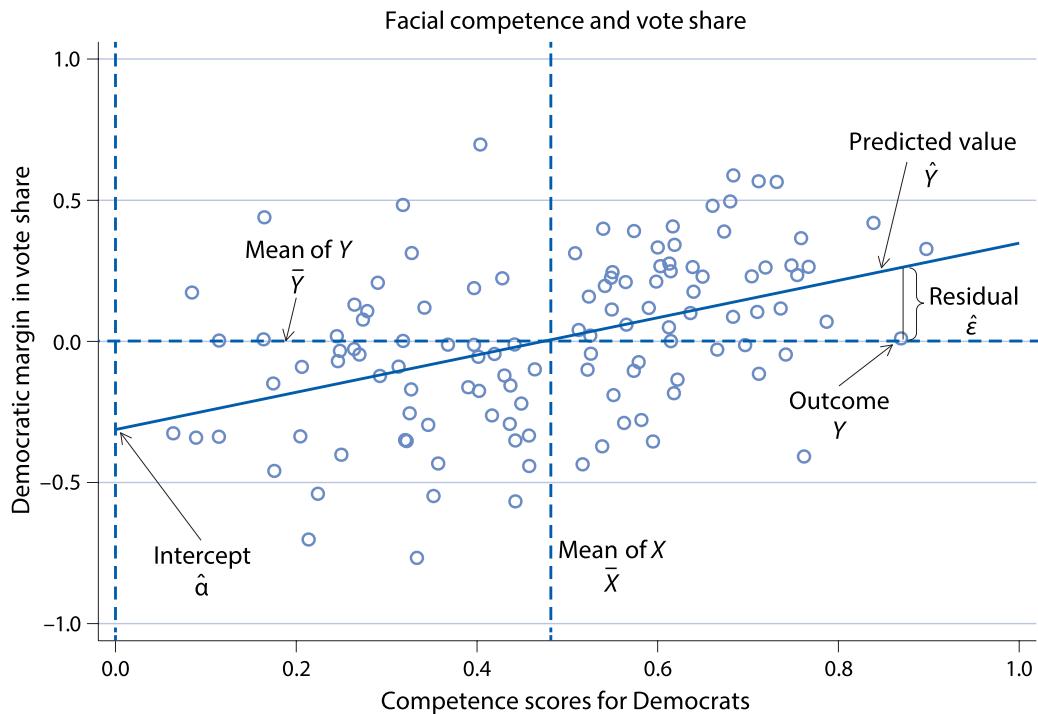


Figure 4.4. Facial Appearance and Vote Shares.

It is straightforward to add the regression line to the scatterplot by replacing the `regress` command with `lfit` in our graph code, specifying our dependent variable and one independent variable. If we want to plot the fit from a regression with multiple independent variables, we could run the regression, use `predict` to calculate the predicted values, and then use the `line` command to plot them. 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. Below we provide the syntax for a simpler version of figure 4.4, where we added additional information to better demonstrate how the concepts we have presented relate to one another.

```
scatter diffshare dcomp, msymbol(Oh) || lfit diffshare dcomp, range(0 1) ///
xtitle("Competence scores for Democrats") ///
ytitle("Democratic margin in vote share") ///
title("Facial competence and vote share") legend(off)
```

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

$$\text{SSR} = \sum_{i=1}^n \hat{\epsilon}_i^2 = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^n (Y_i - \hat{\alpha} - \hat{\beta}X_i)^2. \quad (4.4)$$

In the equation,  $Y_i$ ,  $X_i$ , and  $\hat{\epsilon}_i$  represent the outcome variable, the predictor, and the residual, respectively, for the  $i$ th 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

$$\text{RMSE} = \sqrt{\frac{1}{n} \text{SSR}} = \sqrt{\frac{1}{n} \sum_{i=1}^n \hat{\epsilon}_i^2}. \quad (4.5)$$

Therefore, RMSE represents the average magnitude of the prediction error for the regression, and this is what the method of least squares minimizes.

In Stata, the RMSE is provided in the regression output in the top right-hand corner. However, just like results from general commands, such as `summarize`, are stored in `r()` (e.g., `r(mean)`), we can access results from estimation commands, such as `regress`, using `e()`. Like `r` results, `e()` will always store the results of the most recently run estimation command. To view the RMSE of our last regression, we simply use the following:

```
. display e(rmse)
.26648493
```

While Stata automatically includes this calculation in the regression output, we can alternatively obtain the residuals using the `predict` command with the `residuals` option. We would then compute the RMSE by taking the square root of the squared residuals, as calculated in section 4.1.3,

```
. predict resid, residuals
. generate resid2 = resid^2
. summarize resid2, meanonly
. display sqrt(r(mean))
.26423608
```

The difference between the calculated RMSE and the one given by Stata is that Stata's calculation accounts for the 2 degrees of freedom used in the regression. We discuss degrees of freedom later in this chapter. Overall, 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 approximately 26 percentage points.

The least squares estimates of intercept and slope parameters are given by

$$\hat{\alpha} = \bar{Y} - \hat{\beta}\bar{X} \quad (4.6)$$

$$\hat{\beta} = \frac{\sum_{i=1}^n (Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}. \quad (4.7)$$

Recall that the sample mean of  $Y$  and  $X$  are given by  $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$  and  $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$ , respectively. The results imply that the regression line always goes through the center of the data  $(\bar{X}, \bar{Y})$ . This is so because substituting  $x = \bar{X}$  into equation (4.2) and using the expression for  $\hat{\alpha}$  in equation (4.6) yields  $\hat{Y} = \bar{Y}$ ,

$$\hat{Y} = \underbrace{(\bar{Y} - \hat{\beta}\bar{X})}_{\hat{\alpha}} + \hat{\beta}\bar{X} = \bar{Y}.$$

In figure 4.4, 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:

$$\text{mean of } \hat{\epsilon} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{\alpha} - \hat{\beta}X_i) = \bar{Y} - \hat{\alpha} - \hat{\beta}\bar{X} = 0.$$

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,

$$\text{SSR} = \sum_{i=1}^n \hat{\epsilon}_i^2 = \sum_{i=1}^n (Y_i - \hat{\alpha} - \hat{\beta}X_i)^2.$$

The mean of residuals is always zero, and the regression line always goes through the center of data  $(\bar{X}, \bar{Y})$  where  $\bar{X}$  and  $\bar{Y}$  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:

$$\begin{aligned} \hat{\beta} &= \frac{1}{n} \sum_{i=1}^n \frac{(Y_i - \bar{Y})(X_i - \bar{X})}{\sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2} \sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2}} \times \frac{\sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2}} \\ &= \text{correlation of } X \text{ and } Y \times \frac{\text{standard deviation of } Y}{\text{standard deviation of } X}. \end{aligned} \quad (4.8)$$

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.,  $\sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2}$ , 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  $\rho$  standard deviations in  $Y$ , where  $\rho$  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 perceived competence score and the two-party vote share differential is .43 whereas the standard deviations of  $X$  and  $Y$  are .19 and .29, respectively. Thus, an increase in the perceived competence score of .19 is associated with an average increase of the two-party vote share differential of roughly 12 percentage points ( $\approx .43 \times .29$ ).

The estimated **slope coefficient** from a linear regression model equals the  $\rho$  standard deviation unit increase in the outcome variable that is associated with an increase of 1 standard deviation in the predictor, where  $\rho$  is the correlation between the two variables.

#### 4.2.4 REGRESSION TOWARD 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 4.5 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 4.5, 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.

Figure 4.5. 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.

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, and show in chapter 5, that this empirical phenomenon can be explained by chance alone. Thus, regression toward the mean does not imply that human heights are converging and everyone will have an identical height in the future!

Regression toward the mean is observed in other contexts as well. 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 toward 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.

**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

We will examine whether or not the US presidential election data exhibit the regression toward 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.dta`, into the 2008 election data set. The variable names and descriptions of the 2012 election result data set are given in table 4.5. We will use the state name variable `state`, which is contained in both data sets, to merge the two data files.

Because some variables have the same name in both data sets despite differing in years, we add a year suffix to the `obama`, `romney`, and `ev` variables in 2012 and `obama`, `mccain`, and `ev` variables in 2008. Using the `rename` command, we place the variables that we would like to rename within the parentheses followed by an equals sign and the suffix we want to add. Note that there should be no space between the equals sign and suffix or Stata will display an error message. We save the updated 2012 data as a temporary file to avoid overwriting the original data. We use the  `tempfile` command to first create a temporary data file called `p12`, to which we then save our modified data and merge into the 2008 data set.

```

. * add suffix and merge two data sets
. use pres12, clear
. rename (obama romney ev) =12
. tempfile p12

```

```

. use pres08, clear
. rename (obama mccain ev) =08
. merge 1:1 state using `p12'
      Result          # of obs.
      not matched          0
      matched            51  (_merge==3)

```

Using the merged data set, we investigate whether or not the regression toward 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 `std()` function with the `egen` command. We perform this transformation because, technically, the regression toward the mean phenomenon holds when both the outcome and explanatory variables are standardized. Note that the `std()` function should not be confused with the `sd()` function, which provides the standard deviation of a variable.

```
. egen obama08z = std(obama08)
. egen obama12z = std(obama12)
```

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  $\hat{\alpha} = \bar{Y} - \hat{\beta} \bar{X}$  (see equation (4.6)) and after standardizing, the sample means of both variables,  $\bar{Y}$  and  $\bar{X}$ , are zero. As shown below, in this case, Stata estimates the intercept to be essentially zero. It is also possible to fit the model without an intercept by adding the `noconstant` option to the end of the `regress` command.

```
. * intercept is estimated as essentially zero
. regress obama12z obama08z

      Source |       SS           df          MS      Number of obs   =     51
              |                                 F(1, 49)    =  1442.99
Model      |  48.357896            1  48.357896  Prob > F    =  0.0000
Residual   |  1.64210452          49  .033512337 R-squared     =  0.9672
              |                               Adj R-squared =  0.9665
Total      |  50.00000005        50  1.00000001  Root MSE    =  .18306

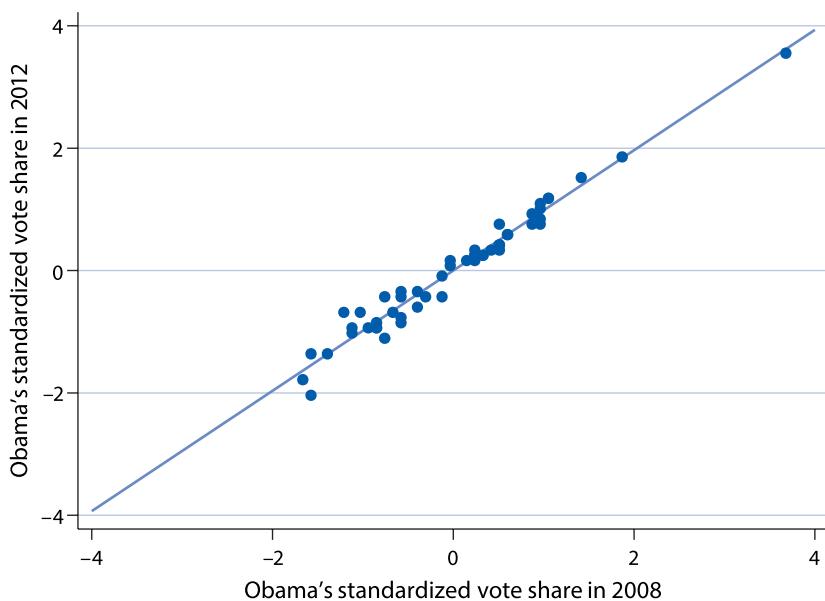
obama12z |      Coef.      Std. Err.          t      P>|t| [95% Conf. Interval]
obama08z |  .9834419  .0258891      37.99  0.000  .9314157  1.035468
_cons    |  3.96e-09  .0256341       0.00  1.000  -.0515136  .0515136

.
. * regression without an intercept; estimated slope is identical
. regress obama12z obama08z, noconstant
```

Source	SS	df	MS	Number of obs	=	51
Model	48.357896	1	48.357896	F(1, 50)	=	1472.44
Residual	1.64210452	50	.03284209	Prob > F	=	0.0000
				R-squared	=	0.9672
				Adj R-squared	=	0.9665
Total	50.0000005	51	.980392167	Root MSE	=	.18122
<hr/>						
obama12z	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
obama08z	.9834419	.0256289	38.37	0.000	.9319647	1.034919

We plot the fitted regression line as well as the data points where we observe a strong linear relationship.

```
. scatter obama12z obama08z, || lfit obama08z obama12z, ///
> xlabel(-4(2)4) ylabel(-4(2)4) range(-4 4) ///
> xtitle("Obama's standardized vote share in 2008") ///
> ytitle("Obama's standardized vote share in 2012") legend(off)
```



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 toward the mean phenomenon exists, then this proportion should be greater for the states in the bottom quartile than those in the top quartile.

In the code that follows, we use the `cond()` function to create a dichotomous variable where 1 (0) indicates Obama's 2012 vote share being greater than (less than or equal to)

his 2008 vote share. Then we obtain the bottom and top quartiles using the `summarize` command with the `detail` option. The quartile values are stored respectively in `r(p25)` and `r(p75)`. We save these values as scalars because we use them in subsequent `summarize` commands that would otherwise replace the results stored from the initial command. The latter two `summarize` commands indicate whether Obama's 2008 vote share for a state is in the bottom or top quartile. We save the data set for use in the next section.

```

. * create dichotomous variable
. generate obama12w = cond(obama12z > obama08z, 1, 0)

.
. * save quartile values
. summarize obama08z, detail

      Standardized values of (obama08)



|     | Percentiles | Smallest  |             |           |
|-----|-------------|-----------|-------------|-----------|
| 1%  | -1.664202   | -1.664202 |             |           |
| 5%  | -1.573621   | -1.573621 |             |           |
| 10% | -1.120716   | -1.573621 | Obs         | 51        |
| 25% | -.7583929   | -1.392459 | Sum of Wgt. | 51        |
| 50% | -.0337458   |           | Mean        | -7.74e-09 |
|     |             | Largest   | Std. Dev.   | 1         |
| 75% | .6003204    | 1.053225  |             |           |
| 90% | .9626439    | 1.415548  | Variance    | 1         |
| 95% | 1.415548    | 1.868453  | Skewness    | .849254   |
| 99% | 3.680071    | 3.680071  | Kurtosis    | 4.918284  |


. scalar p25 = r(p25)
. scalar p75 = r(p75)

.
. * bottom quartile
. summarize obama12w if obama08z <= p25



| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| obama12w | 14  | .5714286 | .5135526  | 0   | 1   |


.
. * top quartile
. summarize obama12w if obama08z >= p75



| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| obama12w | 13  | .4615385 | .5188745  | 0   | 1   |


.
. save pres0812.dta, replace
file pres0812.dta saved

```

The results clearly show the regression toward 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.

#### 4.2.5 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

$$\text{TSS} = \sum_{i=1}^n (Y_i - \bar{Y})^2.$$

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 Stata file `florida.dta` 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 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 calculating the TSS and then SSR. Recall that the `predict` command with the `residuals` option calculates the residuals from the regression output, which we store in `resid1`. We also store the predicted values in variable `fitted`, using the `xb` option.

**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

.	* load data and regress Buchanan's 2000 votes on Perot's 1996 votes					
.	use florida, clear					
.	regress buchanan00 perot96					
<hr/>						
Source	SS	df	MS	Number of obs	=	67
Model	6854381.12	1	6854381.12	F(1, 65)	=	68.48
Residual	6506117.66	65	100094.118	Prob > F	=	0.0000
Total	13360498.8	66	202431.8	R-squared	=	0.5130
				Adj R-squared	=	0.5055
				Root MSE	=	316.38
<hr/>						
buchanan00	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
perot96	.035915	.0043401	8.28	0.000	.0272473	.0445828
_cons	1.345752	49.75931	0.03	0.979	-98.03045	100.7219
<hr/>						

```
.
. * obtain predicted values and residuals
. predict fitted, xb
. predict resid1, residuals
.
. * obtain mean votes for Buchanan
. summarize buchanan00
```

Variable	Obs	Mean	Std. Dev.	Min	Max
buchanan00	67	260.6716	449.9242	9	3407

```
.
. * compute TSS (total sum of squares)
. generate tss1 = (buchanan00 - r(mean))^2
. summarize tss1

Variable | Obs Mean Std. Dev. Min Max
tss1 | 67 199410.4 1205964 5.421252 9899382

. scalar tss_sum = r(sum)

.

. * compute SSR (sum of squared residuals)
. generate ssr1 = resid1^2
. summarize ssr1

Variable | Obs Mean Std. Dev. Min Max
ssr1 | 67 97106.24 647086.8 .0000142 5297648

. scalar ssr_sum = r(sum)

.

. * coefficient of determination
. display (tss_sum - ssr_sum) / tss_sum
.51303329
```

The result shows that 51% of the variation of Buchanan's 2000 votes can be explained by Perot's 1996 votes. Although it is important to know how to perform these calculations, like the earlier example of RMSE, Stata automatically returns the value of  $R$ -squared ( $R^2$ ) after running the `regress` command. The  $R^2$  result is displayed in the upper-right corner of the regression results table and is stored under `e(r2)` for easy access.

```
. display e(r2)
.51303333
```

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 have to reload the presidential data to run the regression and compute the  $R^2$ , but we will use the `preserve` and `restore` commands so we do not lose our data currently in memory. The coefficient of determination for the Florida regression proves to be much lower than that for the state-level Obama regression.

```

. preserve
. use pres0812, clear
. regress obama12z obama08z

Source |      SS          df         MS      Number of obs =      51
       | 48.357896           1   48.357896      F(1, 49)    = 1442.99
       | 1.64210452          49   .033512337     Prob > F = 0.0000
       |                                         R-squared = 0.9672
       |                                         Adj R-squared = 0.9665
       | 50.00000005          50   1.00000001     Root MSE = .18306
       |
obama12z |      Coef.    Std. Err.          t      P>|t| [95% Conf. Interval]
       |
obama08z |  .9834419   .0258891      37.99  0.000   .9314157  1.035468
_cons   |  3.96e-09   .0256341      0.00  1.000  -.0515136  .0515136
       |

. display e(r2)
.96715791
. restore

```

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. Stata can graph these values by running `rvfplot` after a `regress` command. We add a horizontal line at zero using the `yline` option.

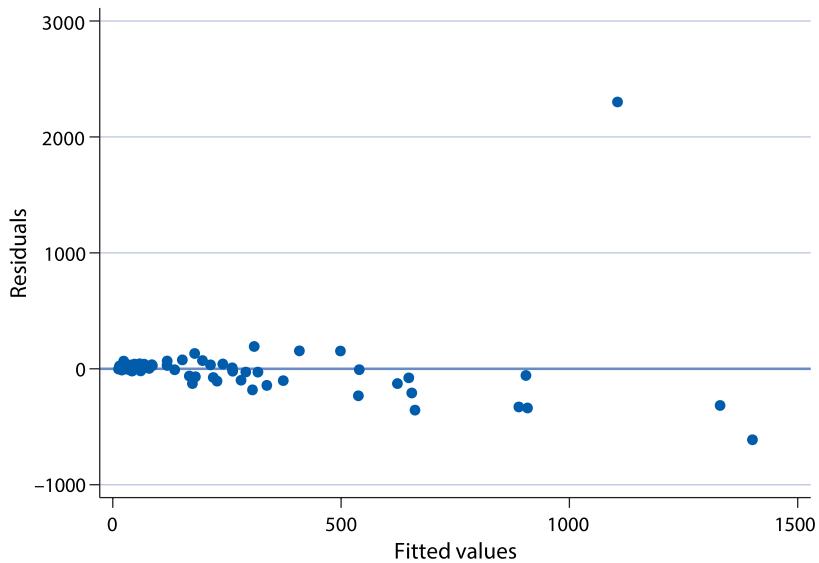
```

. regress buchanan00 perot96

Source |      SS          df         MS      Number of obs =      67
       | 6854381.12           1   6854381.12      F(1, 65)    = 68.48
       | 6506117.66          65   100094.118     Prob > F = 0.0000
       |                                         R-squared = 0.5130
       |                                         Adj R-squared = 0.5055
       | 13360498.8          66   202431.8     Root MSE = 316.38
       |
buchenan00 |      Coef.    Std. Err.          t      P>|t| [95% Conf. Interval]
       |
perot96  |  .035915   .0043401      8.28  0.000   .0272473  .0445828
_cons   |  1.345752   49.75931      0.03  0.979  -98.03045  100.7219
       |

. * built-in graph command
. rvfplot, yline(0)

```



Though not displayed, we could also use the following syntax to plot the graph ourselves using the fitted value and residual variables we created earlier.

```
* using created variables
scatter resid1 fitted, yline(0)
```

We observe an extremely large residual or *outlier*, where in the 2000 election, Buchanan received over 2,000 votes, substantially more than expected. The `summarize` and `list` commands that follow show that this observation represents Palm Beach county. This can be seen by extracting the county name whose residual equals the maximum value of residuals. In this case, the expression `resid1 == r(max)` works to extract the county name.

```
. summarize resid1
      Variable   Obs    Mean    Std. Dev.    Min    Max
      resid1     67    2.46e-07    313.9706   -612.7394   2301.662
. list county if resid1 == r(max)
      county
50.  PalmBeach
```

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 4.6. 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

Figure 4.6. Butterfly Ballot in Palm Beach County. *Source:* Wikipedia

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.

```
. * regression without Palm Beach
. regress buchanan00 perot96 if county != "PalmBeach"

      Source |       SS           df          MS      Number of obs =       66
              |   2818322.7           1    2818322.7      F(1, 64)      =  366.01
              |   492803.298          64    7700.05153     Prob > F      = 0.0000
              |                                         R-squared      =  0.8512
              |                                         Adj R-squared =  0.8488
              |   3311126            65    50940.4      Root MSE      =  87.75

      buchanan00 |      Coef.        Std. Err.          t      P>|t| [95% Conf. Interval]
              |   .0243522     .0012729     19.13    0.000     .0218093     .026895
              |   45.84193    13.89275      3.30    0.002    18.08798    73.59588

.
. * R-squared or coefficient of determination
. display e(r2)
.85116746
```

Without Palm Beach, the coefficient of determination dramatically increases from .51 to .85. The improvement in model fit can also be easily seen through the residual plot as well as the scatterplot 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.

```
. * predicted values and residual, excluding Palm Beach
. predict xb_nopb, xb

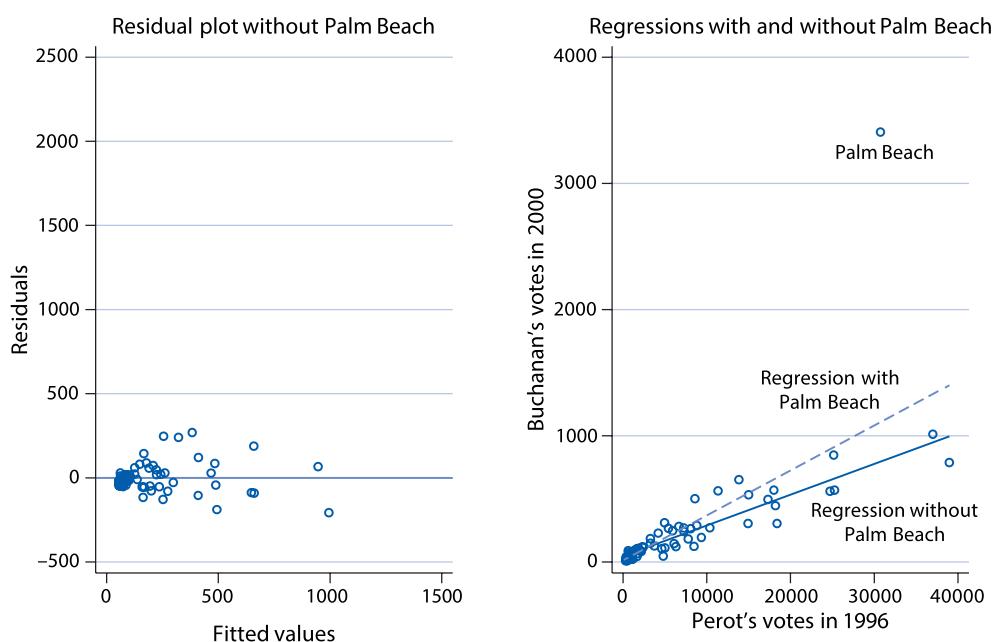
. predict resid_nopb, residuals

.

. * residual plot
. scatter resid_nopb xb_nopb if county!="PalmBeach", yline(0) ///
>         xlabel(0(500)1500) ylabel(-500(500)2500) ///
>         xtitle("Fitted values") ytitle("Residuals") ///
>         title("Residual plot without Palm Beach", size(medsmall)) ///
>         msymbol(Oh) name(resid, replace)

. * regression lines with and without Palm Beach
. scatter buchanan00 perot96, msymbol(Oh) || ///
>         lfit buchanan00 perot96, lpattern(dash) || ///
>         lfit buchanan00 perot96 if county!="PalmBeach", ///
>         xtitle("Perot's votes in 1996") ytitle("Buchanan's votes in 2000") ///
>         title("Regressions with and without Palm Beach", size(medsmall)) ///
>         text(3250 30000 "Palm Beach") ///
>         text(1500 30000 "Regression with" "Palm Beach") ///
>         text(300 30000 "Regression without" "Palm Beach") ///
>         legend(off) name(fit, replace)

. graph combine resid fit
```



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 that 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.

### 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.<sup>5</sup> 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 a 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 data set `women.dta` contains a subset of these 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. We obtain the average for each type of GP using `tabulate` with the `summarize()` option.

<sup>5</sup>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.

**Table 4.7.** Women as Policymakers Data.

Variable	Description
go	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

```
. use women, clear
. * proportion of female politicians in unreserved GP vs. reserved GP
. tabulate reserved, summarize(female)
```

reserved	Summary of female		
	Mean	Std. Dev.	Freq.
0	.07476636	.26363065	214
1	1	0	108
Total	.38509317	.48737471	322

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 are concerned with the quality of drinking water than men, who are more frequently concerned 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
. summarize water if reserved == 1, meanonly
. scalar reservel = r(mean)
```

```

. summarize water if reserved == 0, meanonly
. scalar reserve0 = r(mean)
. display reservel - reserve0
9.252423

.
. * irrigation facilities
. summarize irrigation if reserved == 1, meanonly
. scalar irrigl = r(mean)
. summarize irrigation if reserved == 0, meanonly
. scalar irrig0 = r(mean)
. display irrigl - irrig0
-.36933195

```

We find that the reservation policy increased the number of drinking water facilities in a GP on average by about 9 (new or repaired), raising the mean from approximately 15 to 24 facilities. In contrast, 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. Consequently,  $\hat{\beta}$  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*.

```
. regress water reserved
```

Source	SS	df	MS	Number of obs	=	322
Model	6144.58583	1	6144.58583	F(1, 320)	=	5.49
Residual	357956.337	320	1118.61355	Prob > F	=	0.0197
				R-squared	=	0.0169
Total	364100.922	321	1134.27079	Adj R-squared	=	0.0138
				Root MSE	=	33.446
water	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reserved	9.252423	3.947746	2.34	0.020	1.485608	17.01924
_cons	14.73832	2.2863	6.45	0.000	10.24024	19.2364

```
. regress irrigation reserved
```

Source	SS	df	MS	Number of obs	=	322
Model	9.79073762	1	9.79073762	F(1, 320)	=	0.11
Residual	28914.7714	320	90.3586605	Prob > F	=	0.7422
				R-squared	=	0.0003
Total	28924.5621	321	90.1076701	Adj R-squared	=	-0.0028
				Root MSE	=	9.5057
irrigation	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reserved	-.3693319	1.122003	-0.33	0.742	-2.576766	1.838103
_cons	3.38785	.6497975	5.21	0.000	2.109436	4.666265

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  $\widehat{Y}(1) - \widehat{Y}(0) = (\widehat{\alpha} + \widehat{\beta}) - \widehat{\alpha} = \widehat{\beta}$ , while the estimated average outcome for the control group is equal to the estimated intercept, i.e.,  $\widehat{Y}(0) = \widehat{\alpha}$ . 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

the 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,  $\alpha$  is the intercept,  $\beta_j$  is the coefficient for predictor  $X_j$ ,  $\epsilon$  is an error term, and  $p$  is the number of predictors and can be greater than 1. The interpretation of each coefficient  $\beta_j$  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  $(\hat{\alpha}, \hat{\beta}_1, \dots, \hat{\beta}_p)$  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  $j$ th predictor for the  $i$ th observation. Recall that the residual is defined as the difference between the observed response  $Y$  and its predicted or fitted value  $\hat{Y} = \hat{\alpha} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$ .

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 `regress` command as before. One can add more than one predictor by simply adding new variable names. In this example, since `messages` is a factor variable, we can add the prefix `i.` before it to automatically create 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 set. The model includes all but the variable corresponding to the base level. The base level of a factor variable is the lowest value of the variable. The other values of a factor variable are defined in relation to this base level value. Here, the base level of the messages variable corresponds to the Civic Duty condition (i.e., messages == 1).

. use social, clear						
. regress primary2008 i.messages						
Source	SS	df	MS	Number of obs	=	305,866
Model	215.612203	3	71.8707343	F(3, 305862)	=	335.77
Residual	65468.4694	305,862	.214045777	Prob > F	=	0.0000
Total	65684.0816	305,865	.214748603	R-squared	=	0.0033
				Adj R-squared	=	0.0033
				Root MSE	=	.46265
primary2008	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
messages						
Control	-.0178993	.0025923	-6.90	0.000	-.0229801	-.0128186
Hawthorne	.007837	.0033471	2.34	0.019	.0012767	.0143973
Neighbors	.0634106	.0033472	18.94	0.000	.0568501	.069971
_cons	.3145377	.0023666	132.91	0.000	.3098992	.3191761

Alternatively, one can create an indicator variable for each group and then specify the regression model using them. Using tabulate with the generate() option provides us with a dummy for each category of messages. The regression results are identical to those given above.

. * create indicator variables						
. tabulate messages, generate(message)						
messages	Freq.	Percent	Cum.			
Civic Duty	38,218	12.50	12.50			
Control	191,243	62.53	75.02			
Hawthorne	38,204	12.49	87.51			
Neighbors	38,201	12.49	100.00			
Total	305,866	100.00				

. regress primary2008 message2 message3 message4						
Source	SS	df	MS	Number of obs	=	305,866
Model	215.612203	3	71.8707343	F(3, 305862)	=	335.77
Residual	65468.4694	305,862	.214045777	Prob > F	=	0.0000
				R-squared	=	0.0033
Total	65684.0816	305,865	.214748603	Adj R-squared	=	0.0033
				Root MSE	=	.46265
primary2008	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
message2	-.0178993	.0025923	-6.90	0.000	-.0229801	-.0128186
message3	.007837	.0033471	2.34	0.019	.0012767	.0143973
message4	.0634106	.0033472	18.94	0.000	.0568501	.069971
_cons	.3145377	.0023666	132.91	0.000	.3098992	.3191761

Mathematically, the linear regression model we just fit is given by

$$Y = \alpha + \beta_1 \text{Control} + \beta_2 \text{Hawthorne} + \beta_3 \text{Neighbors} + \epsilon.$$

In this model, each predictor is an indicator variable for the corresponding group, where the dummy for the base level `Civic Duty` condition is omitted. 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  $\hat{\alpha} + \hat{\beta}_1 = .315 + (-.018) = .297$  or 29.7%. Similarly, for the `Neighbors` group, the predicted average outcome is  $\hat{\alpha} + \hat{\beta}_3 = .315 + .063 = .378$ .

The predicted average outcome can be obtained using the `predict` command. As reviewed, this command takes the output from the `regress` command and computes predicted values. It can make predictions in-sample (using only the observations from the model), out-of-sample (using all possible observations), or using other data sets. The new data set'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 set using the `duplicates drop` command. The resulting data set contains the same variables 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 `preserve` and `restore` commands to take a snapshot of the data before dropping duplicates and then return the data to its preserved condition, discarding any changes made in the interim.

```
. preserve
.
    * create a data set with unique values of messages
.
    duplicates drop messages, force
```

```
Duplicates in terms of messages
(305,862 observations deleted)

. tabulate messages



| messages   | Freq. | Percent | Cum.   |
|------------|-------|---------|--------|
| Civic Duty | 1     | 25.00   | 25.00  |
| Control    | 1     | 25.00   | 50.00  |
| Hawthorne  | 1     | 25.00   | 75.00  |
| Neighbors  | 1     | 25.00   | 100.00 |
| Total      | 4     | 100.00  |        |



.

. * make prediction for each observation from this new data set
. predict newfit, xb
. tabulate messages, summarize(newfit)



| messages  | Summary of Linear prediction |           |       |
|-----------|------------------------------|-----------|-------|
|           | Mean                         | Std. Dev. | Freq. |
| Civic Dut | .31453764                    | 0         | 1     |
| Control   | .29663831                    | 0         | 1     |
| Hawthorne | .32237461                    | 0         | 1     |
| Neighbors | .37794822                    | 0         | 1     |
| Total     | .3278747                     | .03507705 | 4     |



. restore
```

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
. tabulate messages, summarize(primary2008)



| messages  | Summary of primary2008 |           |         |
|-----------|------------------------|-----------|---------|
|           | Mean                   | Std. Dev. | Freq.   |
| Civic Dut | .31453765              | .46433755 | 38,218  |
| Control   | .29663831              | .45677687 | 191,243 |
| Hawthorne | .32237462              | .46739164 | 38,204  |
| Neighbors | .37794822              | .48488093 | 38,201  |
| Total     | .31224458              | .46340976 | 305,866 |


```

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 include `noconstant` as an option. For the `messages` variable, we also specify that there is no base group by adding `ibn.` as a prefix.

<pre>. * linear regression without intercept . regress primary2008 ibn.messages, noconstant</pre>						
Source	SS	df	MS	Number of obs	=	305,866
Model	30036.5306	4	7509.13265	F(4, 305862)	=	35081.90
Residual	65468.4694	305,862	.214045777	Prob > F	=	0.0000
Total	95505	305,866	.312244578	R-squared	=	0.3145
				Adj R-squared	=	0.3145
				Root MSE	=	.46265
<hr/>						
primary2008	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
messages						
Civic Duty	.3145377	.0023666	132.91	0.000	.3098992	.3191761
Control	.2966383	.0010579	280.39	0.000	.2945648	.2987118
Hawthorne	.3223746	.002367	136.20	0.000	.3177354	.3270139
Neighbors	.3779482	.0023671	159.67	0.000	.3733088	.3825877

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  $.378 - .297$  in the model with no intercept, or  $.063 - (-.018)$  in the original model, either of which equals  $.081$  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. We can access the stored estimates from our last regression by using the system variable `_b` followed by the variable name in square brackets. To obtain the stored coefficients for the Control and Neighbors treatments, we use the variable value as a prefix (e.g., `_b[2.messages]` holds the estimate for the Control group dummy).

```

. * estimated average effect of "Neighbors" condition
. display _b[4.messages] - _b[2.messages]
.08130991

.
. * difference-in-means
. summarize primary2008 if messages == 4, meanonly
. scalar neigh = r(mean)
. summarize primary2008 if messages == 2, meanonly
. scalar control = r(mean)
. display neigh - control
.08130991

```

Finally, we can compute the *coefficient of determination* or  $R^2$  as in section 4.2.5. 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, 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  $R^2$  downward 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. We can obtain both the adjusted and unadjusted  $R^2$  through the command `ereturn list` after quietly rerunning our original regression. The unadjusted  $R^2$  is stored as `e(r2)` while the adjusted  $R^2$  is stored as `e(r2_a)` (see section 6.3).

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.

```

. quietly regress primary2008 i.messages
. display e(r2)
.00328256

. display e(r2_a)
.00327279

```

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  $\beta_j$  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
. summarize primary2008 if messages == 4 & primary2004 == 1, meanonly
. scalar voteneigh = r(mean)
. summarize primary2008 if messages == 2 & primary2004 == 1, meanonly
. scalar votecontrol = r(mean)
. scalar atevoter = voteneigh - votecontrol
. display atevoter
.09652525

.
. * average effect among those who did not vote
. summarize primary2008 if messages == 4 & primary2004 == 0, meanonly
. scalar nonneigh = r(mean)
. summarize primary2008 if messages == 2 & primary2004 == 0, meanonly
. scalar noncontrol = r(mean)
. scalar atenonvoter = nonneigh - noncontrol
. display atenonvoter
.06929617

```

```

.
. * difference
. display atevoter - atenonvoter
.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

$$Y = \alpha + \beta_1 \text{primary2004} + \beta_2 \text{Neighbors} + \beta_3 (\text{primary2004} \times \text{Neighbors}) + \epsilon. \quad (4.9)$$

The final predictor is the product of two indicator variables, `primary2004`  $\times$  `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`).

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  $\beta_2$ . Thus, the coefficient for the interaction term  $\beta_3$  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  $\beta_3$  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  $\beta_1$  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  $\beta_3$ .

In Stata, an interaction term can be represented by a double hashtag `#` with the syntax `x1##x2` producing an interaction term between the two variables `x1` and `x2` and retaining both main effects in the model. We illustrate the use of interaction terms by focusing on the Neighbors and Control groups.

. regress primary2008 i.primary2004##i.messages if messages == 2   messages == 4						
Source	SS	df	MS	Number of obs	=	229,444
Model	1510.89127	3	503.630424	F(3, 229440)	=	2428.48
Residual	47582.51	229,440	.207385417	Prob > F	=	0.0000
Total	49093.4013	229,443	.213967745	R-squared	=	0.0308
				Adj R-squared	=	0.0308
				Root MSE	=	.4554
primary2008	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
1.primary2004	.1486951	.0021253	69.96	0.000	.1445295	.1528607
messages						
Neighbors	.0692962	.0033103	20.93	0.000	.0628081	.0757843
primary2004#messages						
1#Neighbors	.0272291	.0051977	5.24	0.000	.0170417	.0374165
_cons	.2371099	.0013448	176.32	0.000	.2344742	.2397456

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, a single hashtag `#` can be used but the main effects have to be entered separately. If the main effects are not included and a single hashtag is used with categorical variables, Stata will show indicators for each combination of the variables. When used with continuous variables, Stata will include only the interaction term. In most applications, one

should include the corresponding main effects when the model has an interaction term. The same regression model can be fitted using the following syntax.

```
. regress primary2008 i.primary2004 i.messages i.primary2004#i.messages ///
>      if messages == 2 | messages == 4
```

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  $\beta_3$  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 2008 as a predictor. We first compute this variable by subtracting the year of birth variable from the year of election.

```
. generate age = 2008 - yearofbirth
. summarize age
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age	305,866	51.78558	14.4533	22	108

In this subset of the data, the age of voters varies from 22 to 108. 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 before 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  $\{\hat{\alpha} + \hat{\beta}_1(x+1) + \hat{\beta}_2 + \hat{\beta}_3(x+1)\} - \{\hat{\alpha} +$

$\hat{\beta}_1(x+1) = \hat{\beta}_2 + \hat{\beta}_3(x+1)$ . Therefore, the estimated coefficient for the interaction effect term  $\hat{\beta}_3 = \{\hat{\beta}_2 + \hat{\beta}_3(x+1)\} - (\hat{\beta}_2 + \hat{\beta}_3x)$  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 Stata, 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. We add the prefix `c.` to `age` rather than `i.` to let Stata know that it is a continuous variable, not an indicator variable.

. regress primary2008 c.age##i.messages if messages == 2   messages == 4						
Source	SS	df	MS	Number of obs	=	229,444
Model	1021.43392	3	340.477972	F(3, 229440)	=	1625.05
Residual	48071.9674	229,440	.209518686	Prob > F	=	0.0000
				R-squared	=	0.0208
Total	49093.4013	229,443	.213967745	Adj R-squared	=	0.0208
				Root MSE	=	.45773
primary2008						
	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
age	.0039982	.0000725	55.14	0.000	.0038561	.0041403
messages						
Neighbors	.0485728	.0094908	5.12	0.000	.0299711	.0671745
messages#c.age						
Neighbors	.0006283	.0001762	3.57	0.000	.0002829	.0009737
_cons	.0894768	.0038998	22.94	0.000	.0818334	.0971203

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 .06 percentage points. Based on this regression model, we can also compute the estimated average treatment effect for different ages. The `margins` command enables us to calculate predicted values under hypothetical specifications. In other words, we set a predictor at a particular value. To illustrate how the treatment effect may vary across ages, we look specifically at the effects at ages 25, 45, 65, and 85. We use the `margins` command after running our regression, and we include the selected ages in the `at()` option. Because we want the difference in the average treatment effect between the two treatment groups, we also specify the `dydx()` option. This option provides the marginal effects for discrete (or categorical) variables, with its name corresponding to the derivative of `y` over the derivative of `x`. We place the grouping variable `messages` inside the parentheses. This shows the change in the effect as we change `messages` from the `Control` to the `Neighbors` condition. The `margins` command makes its calculations based on the most recently run regression.

. * age = 25, 45, 65, 85 in Neighbors group						
. margins, at(age=(25(20)85)) dydx(messages)						
Conditional marginal effects				Number of obs	=	229,444
Model VCE	: OLS					
Expression	: Linear prediction, predict()					
dy/dx w.r.t.	: 4.messages					
1._at	: age	=	25			
2._at	: age	=	45			
3._at	: age	=	65			
4._at	: age	=	85			
<hr/>						
	Delta-method					
	dy/dx	Std. Err.	t	P> t	[95% Conf. Interval]	
2.messages	(base outcome)					
4.messages						
_at						
1	.0642805	.0053821	11.94	0.000	.0537317	.0748293
2	.0768467	.0028348	27.11	0.000	.0712905	.0824028
3	.0894128	.0034575	25.86	0.000	.0826362	.0961895
4	.101979	.0063814	15.98	0.000	.0894716	.1144864

Note: dy/dx for factor levels is the discrete change from the base level.

According to these results, the average treatment effect increases with age, ranging from 6.4 percentage points for individuals aged 25 to more than 10 percentage points for 85-year-olds. However, 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 Stata, the double hashtag ## will not only calculate interactions, but when used between the same variable, it will also include the squared term in the regression. This is important for commands such as margins that calculate standard errors of predictions because the nonsquared and squared variables will vary simultaneously. We now fit the model specified in equation (4.10), which includes the interaction terms.

```
. regress primary2008 (c.age##c.age)##i.messages if messages == 2 | messages == 4
```

Source	SS	df	MS	Number of obs	=	229,444
Model	1151.60662	5	230.321324	F(5, 229438)	=	1102.26
Residual	47941.7947	229,438	.208953158	Prob > F	=	0.0000
Total	49093.4013	229,443	.213967745	R-squared	=	0.0235
				Adj R-squared	=	0.0234
				Root MSE	=	.45711

primary2008	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
age	.0117226	.0003838	30.54	0.000	.0109704	.0124748
c.age#c.age	-.0000739	3.61e-06	-20.49	0.000	-.000081	-.0000668
messages						
Neighbors	-.0527523	.0240982	-2.19	0.029	-.099984	-.0055205
messages#c.age						
Neighbors	.0048043	.0009332	5.15	0.000	.0029753	.0066333
messages#c.age#c.age						
Neighbors	-.0000396	8.74e-06	-4.53	0.000	-.0000567	-.0000225
_cons	-.0969954	.0098972	-9.80	0.000	-.1163936	-.0775973

This model is even more complicated than the last and the coefficients offer no easy interpretation. We therefore, again, use the `margins` command to predict the average outcome under various scenarios. However, here, instead of specifying just four particular ages, we predict the average turnout rate for all voters aged between 25 and 85. To include this entire age range, we place 25/85 within the parentheses for the `at()` option. The slash instructs Stata to include all values starting from 25 until 85. As before, we limit our analysis to the `Neighbors` and `Control` conditions and use the `dydx()` option to get the difference between the two treatment groups.

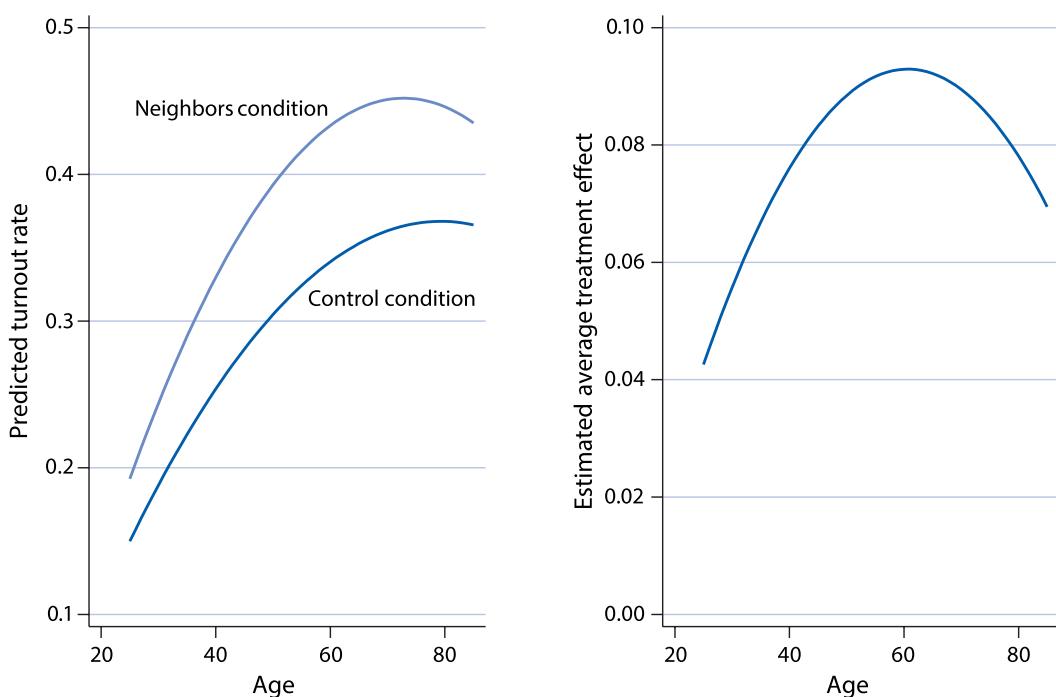
To visualize the results and facilitate interpretation, we use `marginsplot` to plot the effects. We rerun the same regression, suppressing its output using `quietly`. The first plot displays the predicted turnout as a function of age separately for the `Neighbors` and `Control` groups. The second plot shows the estimated average treatment effect as a function of age.

```
. quietly regress primary2008 (c.age##c.age)##i.messages if messages == 2 |
    messages == 4
. * plotting the predicted turnout rate under each condition
```

```

. margins, at(age=(25/85) messages=(2 4))
. marginsplot, recast(line) noci xtitle("Age") ytitle("Predicted turnout rate") ///
>      title("") text(.3 65 "Control condition") ///
>      text(.42 35 "Neighbors condition") ///
>      legend(off) name(margins1, replace)
. * average treatment effect as a function of age
. margins, at(age=(25/85)) dydx(messages)
. marginsplot, recast(line) noci ylabel(0(.02).1) ///
>      title("") xtitle("Age") ytitle("Estimated average treatment effect") ///
>      name(margins2, replace)
. graph combine margins1 margins2

```



One of the nice features of `marginsplot` is that it will plot the confidence interval for us with the option `ci`. In the preceding examples, we turn this option off (with `noci`) because we do not discuss confidence intervals until chapter 6. By default, `marginsplot` will show a scatterplot of the margins values. We use the `recast()` option to instead create a line graph.

The plots indicate 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,

**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)
lngross	log gross wealth at the time of death
lnnet	log net wealth at the time of death
yob	year of birth of the candidate
yod	year of death of the candidate
marginpre	margin of the candidate's party in the previous election
region	electoral region
margin	margin of victory (vote share)

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).

As an application of RD design, we consider how much politicians can increase their personal wealth as a result of holding office. Scholars investigated this question by analyzing members of Parliament (MPs) in the United Kingdom.<sup>6</sup> 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 Stata file `MPs.dta`. 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.

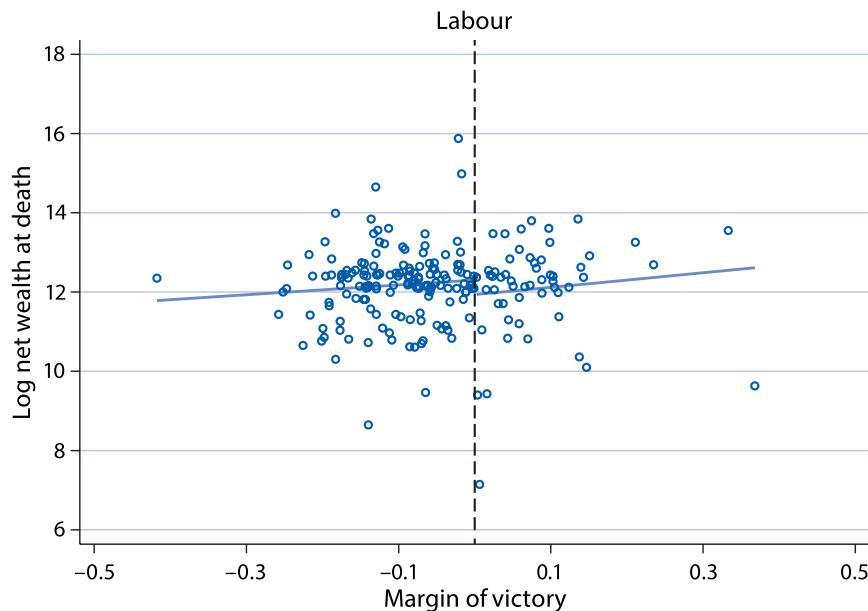
A simple scatterplot with regression lines is the best way to understand RD design. To do this, we plot the outcome variable, log net wealth at the time of death, against the margin of victory. We take the natural logarithmic transformation of wealth because this variable is quite

<sup>6</sup>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.

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). We plot the predicted values from linear regressions using the `lfit` graph command, where we specify our independent and dependent variables as we would with the `regress` command. The difference in predicted values at the point of 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.

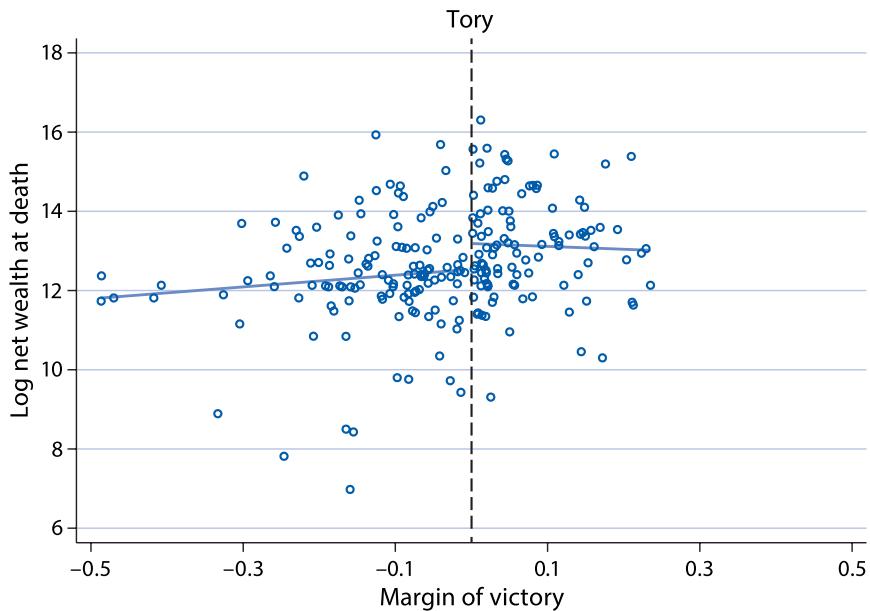
```
. * load the data
. use MPs, clear

. scatter lnnet margin if party == "labour", ///
>     xline(0, lpattern(dash) lcolor(black)) ///
>     xlabel(-.5(.2).5) ylabel(6(2)18) msymbol(oh) || ///
>     lfit lnnet margin if party == "labour" & margin < 0, lcolor(blue) || ///
>     lfit lnnet margin if party == "labour" & margin > 0, lcolor(blue) ///
>     xtitle("Margin of victory") ytitle("Log net wealth at death") ///
>     title("Labour") legend(off) name/labour, replace)
```



```
. scatter lnnet margin if party == "tory", ///
>     xline(0, lpattern(dash) lcolor(black)) ///
>     xlabel(-.5(.2).5) ylabel(6(2)18) msymbol(oh) || ///
```

```
> lfit lnnet margin if party == "tory" & margin < 0, lcolor(blue) || ///
> lfit lnnet margin if party == "tory" & margin > 0, lcolor(blue) ///
> xtitle("Margin of victory") ytitle("Log net wealth at death") ///
> title("Tory") legend(off) name(tory, replace)
```



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 logarithmic function is the exponential function, given by `exp()` in Stata.

In the code that follows, we first rerun separate regressions for those Tory MPs who win the election versus Tory candidates who lost. We then calculate the minimum and maximum vote shares using `summarize` to plug into our `margins` command. We ask Stata to provide the log income when the vote share is held at its minimum in the case of Tory non-MPs and maximum in the case of Tory MPs. As reviewed earlier in this chapter, Stata stores the regression coefficient for the intercept in `_b[_cons]`. We create a new variable to then compute the exponential function for each intercept. The difference between the MP and non-MP variables indicates the causal effect.

```
. * average net wealth for Tory MP
. quietly regress lnnet margin if party == "tory" & margin > 0
. margins, at((max) margin)
Adjusted predictions                               Number of obs      =      102
Model VCE      : OLS
```

```

Expression : Linear prediction, predict()
at        : margin      = .2354484 (max)

+
|   Delta-method
|   Margin Std. Err.      t    P>|t| [95% Conf. Interval]
+
|_cons | 13.01645  .347584  37.45  0.000  12.32685  13.70604

.
. generate torymp = exp(_b[_cons])
.
.
. * average net wealth for Tory non-MP
. quietly regress lnnet margin if party == "tory" & margin < 0
. margins, at((min) margin)

Adjusted predictions                               Number of obs = 121
Model VCE : OLS

Expression : Linear prediction, predict()
at        : margin      = -.486312 (min)

+
|   Delta-method
|   Margin Std. Err.      t    P>|t| [95% Conf. Interval]
+
|_cons | 11.81297  .4763083  24.80  0.000  10.86983  12.7561

.
. generate torynonmp = exp(_b[_cons])
.
.
. * causal effect in pounds
. summarize torymp torynonmp

Variable |   Obs      Mean     Std. Dev.      Min      Max
+
| torymp | 427  533813.4          0  533813.4  533813.4
| torynonmp | 427  278762.5          0  278762.5  278762.5
.
. display torymp - torynonmp
255050.91

```

The estimated effect of being an MP on the personal wealth of Tory candidates is a little above 255,000 pounds. Since the average net wealth for Tory non-MPs is predicted to be a little above 278,000 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
. quietly regress marginpre margin if party == "tory" & margin < 0
. scalar toryneg = _b[_cons]
. quietly regress marginpre margin if party == "tory" & margin > 0
. scalar torypos= _b[_cons]
. * the difference between two intercepts is the estimated effect
. display torypos - toryneg
-.01725578
```

Running separate models for when the margins are negative versus positive in the prior election, we save the coefficients stored in `_b[_cons]` as scalars and calculate the difference. 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 6, 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 toward 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 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,

**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 Democratic nominee's market
pricer	closing price (predicted vote share) of Republican nominee's market
volumed	total session trades of Democratic Party nominee's market
volumer	total session trades of Republican Party nominee's market

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 Stata format as `intrade08.dta` and `intrade12.dta`, 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 found in `pres08.dta` (table 4.1) and `pres12.dta` (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, merge the election data with the market data and keep only the observations that occur the day before the election. 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 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,

**Table 4.10.** 2012 US Presidential Election Polling Data.

Variable	Description
state	abbreviated name of the state in which the poll was conducted
obamapoll	predicted support for Obama (percentage)
romneypoll	predicted support for Romney (percentage)
pollster	name of the organization conducting poll
middate	middate of the period when the poll was conducted

this can be done using time-series operators. This time, we need to specify both a time variable but also an identifier variable (`states`) in our `tsset` command. We first convert the state variable into numeric format, as required by `tsset` using `encode state, generate(newstatevar)`. We then compute the seven-day average within each state using a loop, where we create a variable for each lag value of `obamamargin`. The seven-day average can then be calculated using `egen` with the `rowmean` function. Next, we sum the electoral votes for the states Obama is predicted to win. There will be multiple observations per state for each moving average so we use `collapse` to ensure that we use only one observation for each state and date combination. Plot the results and compare to the previous plot.

4. 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. Merge the intrade data, the polling data, and the election outcome data. Note that the order in which the merges are executed is important and you will want one observation per state and date from the polling data.
5. 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 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.dta`. Table 4.10 presents the names and descriptions of the 2012 US presidential election polling data. You can set up the 2012 files in the same way as the 2008 files and then append the new file to the 2008 data. Compute the predictions based on the Intrade data and plot the predicted statewide election results against the actual results. Examine which states are misclassified. Repeat the analysis using the polling data.

**Table 4.11.** Conditional Cash Transfer Program (Progesa) Data.

Variable	Description
treatment	whether an electoral precinct contains a village where households received early <i>Progesa</i>
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

#### 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 *Progesa*, Mexico's *conditional cash transfer program* (CCT program).<sup>7</sup> 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 *Progesa*) or 6 months (late *Progesa*) 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 in data file `progesa.dta`. Table 4.11 presents the names and descriptions of variables in the data set. Each observation in the data represents

<sup>7</sup>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 Rivera (2020). "Do nonpartisan programmatic policies have partisan electoral effects? Evidence from two large scale randomized experiments." *Journal of Politics*, vol. 81, no. 2, pp. 714–730.

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. 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.
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 better fits the data?
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. Using the `margins` command, estimate the average effects for unique observed values and plot them as a function of the average poverty level. Comment on the resulting plot.

#### 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.<sup>8</sup> 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.dta` 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

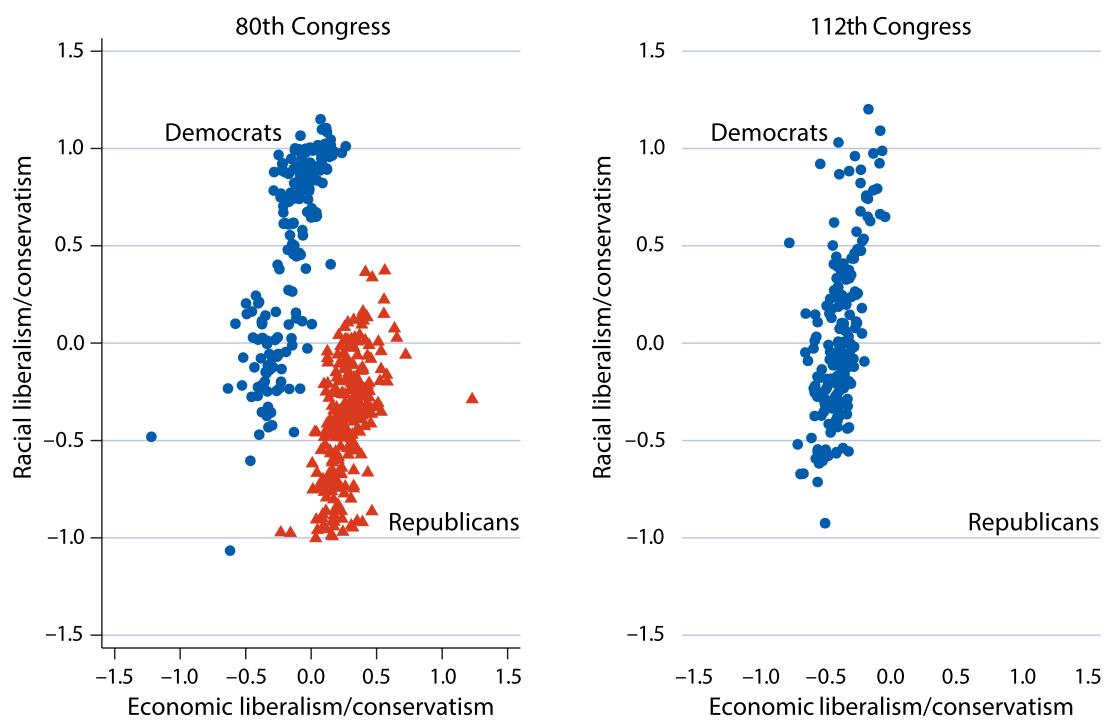
<sup>8</sup>This exercise is based on Stephan Litschig and Kevin M. Morrison (2013). "The impact of intergovernmental transfers on education outcomes and poverty reduction." *American Economic Journal: Applied Economics*, vol. 5, no. 4, pp. 206–240.

**Table 4.12.** Brazilian Government Transfer Data.

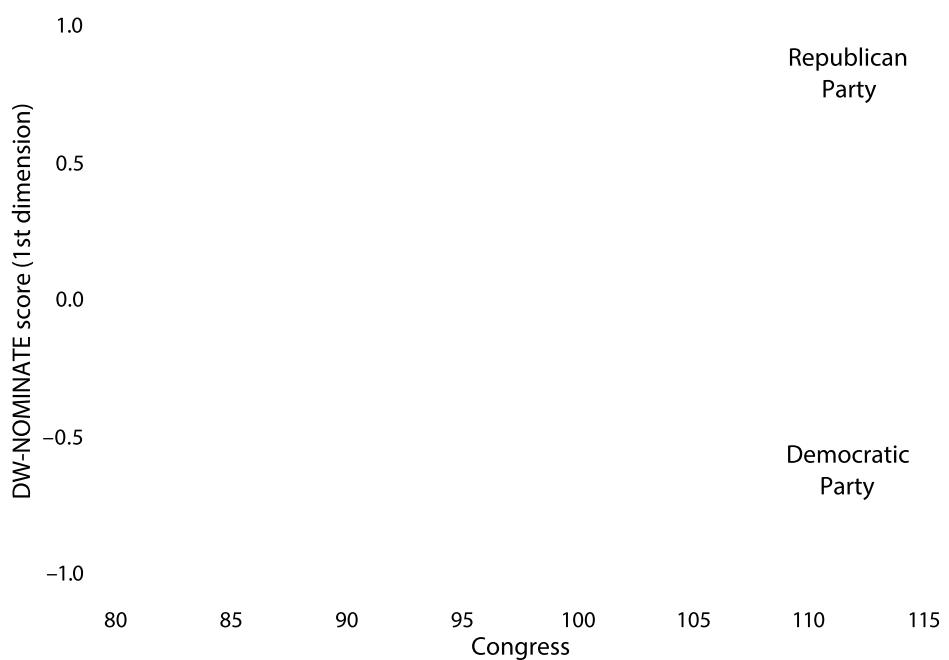
Variable	Description
pop82	population in 1982
poverty80	poverty rate of the state in 1980
poverty91	poverty rate of the state in 1991
educ80	average years in education of the state in 1980
educ91	average years in education of the state in 1991
literate91	literacy rate of the state in 1991
state	state
region	region
id	municipal ID
year	year of measurement

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.

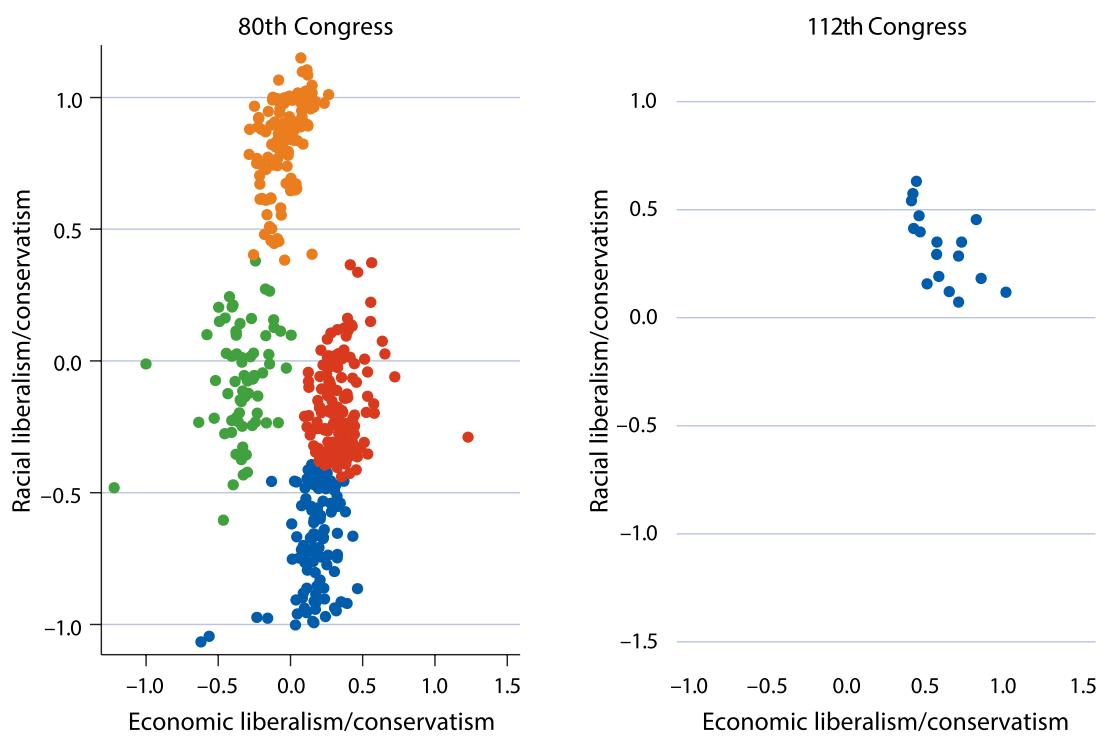
3. For the analysis we will include only those municipalities within three points of the funding cutoff on either side. Using regressions, estimate the average causal effect of government transfer on each of the three outcome variables of interest: educational attainment, literacy, and poverty. Give a brief substantive interpretation of the results.
4. Following the example in section 4.3.4, visualize the analysis done 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 using a loop. 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?



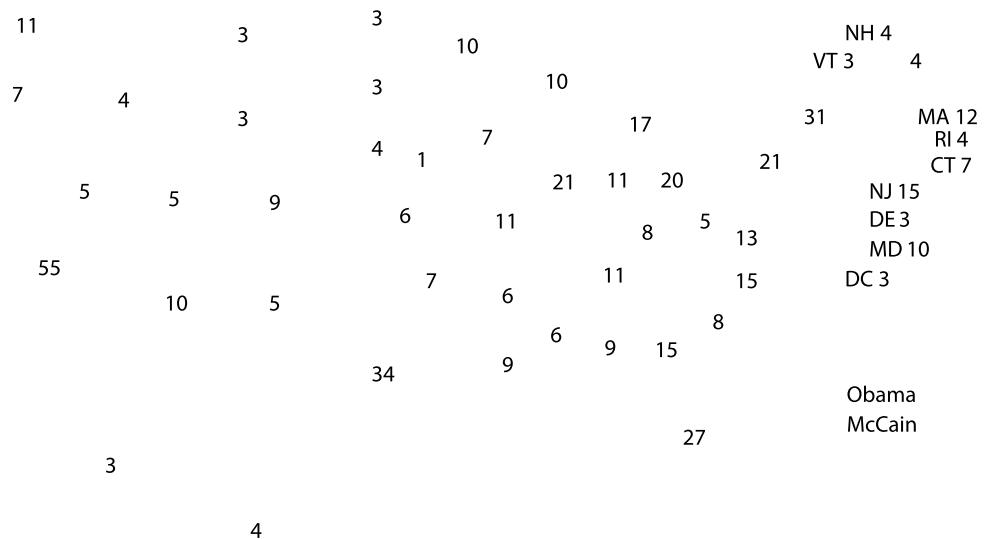
The full-color version of the plots on page 107 in section 3.6.1.



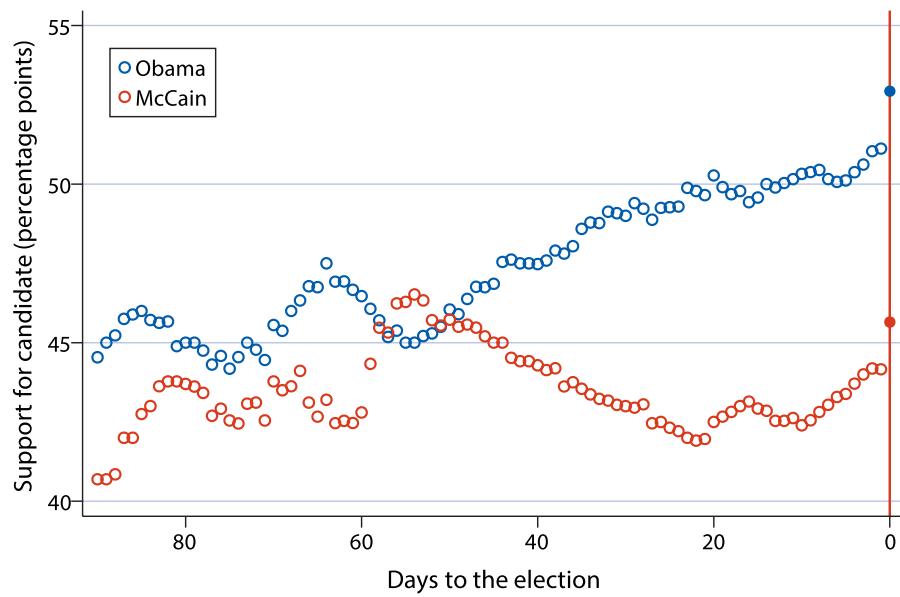
The full-color version of the plot on page 108 in section 3.6.1.



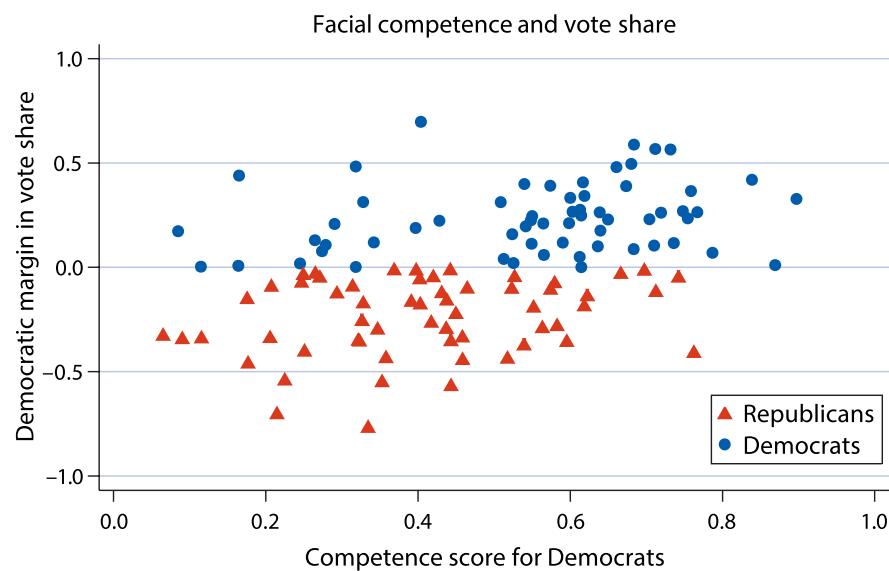
The full-color version of the plots on page 120 in section 3.7.1.



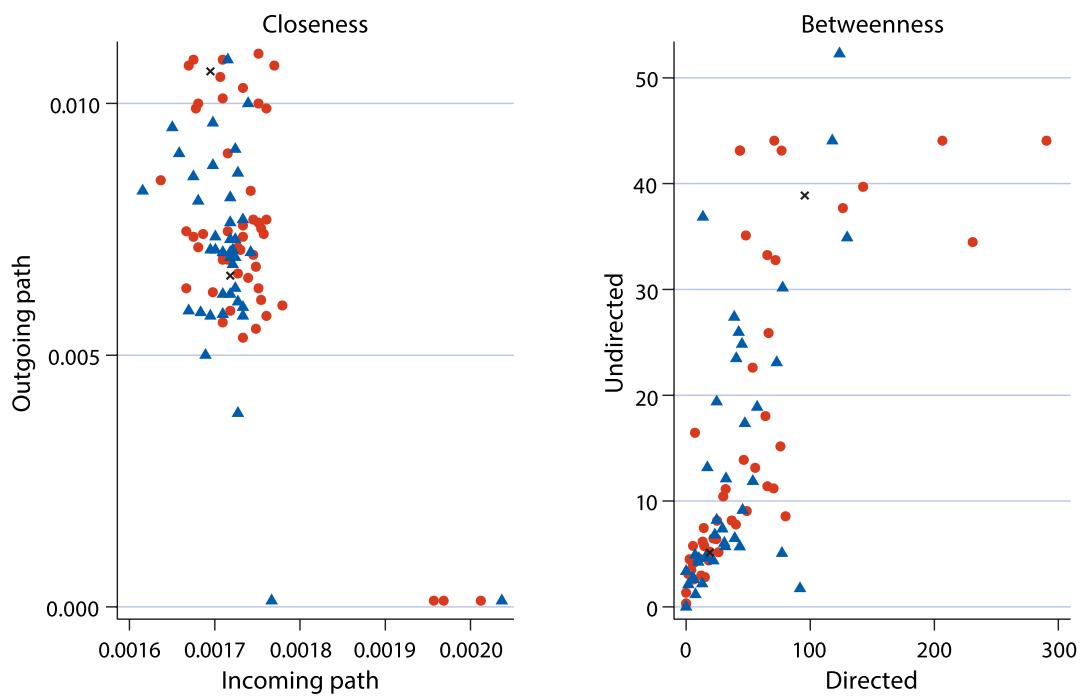
The full-color version of figure 4.1 on page 129.



The full-color version of the plot on page 144 in section 4.1.3.



The full-color version of the plot on page 146 in section 4.2.1.



The full-color version of the plots on page 381 in section 7.1.4.

The full-color version of the plot on page 385 in section 7.1.4.

The full-color version of figure 7.4 on page 388.

US as red and blue states

US as Democratic vote share

The full-color version of the maps on page 395 in section 7.2.3.

Walmart store  
Super Center  
Distribution Center

The full-color version of the map on page 397 in section 7.2.4

1975

1985

1995

2005

The full-color version of the maps on page 398 in section 7.2.5.

Federalist Paper No. 12

land great  
much must  
nation will  
country trade  
far part

commerc  
govern import  
revenu cult collect  
**state**

Federalist Paper No. 24

two must garrison  
peac  
armi power  
upon stand  
object even state

tax appear  
time  
legislatur without necess  
establish nation increas  
**will**

The full-color version of the word cloud on page 408 in section 7.3.2.

## Chapter 5

---

# Probability

Probability is the very guide of life.

—Cicero, *De Natura*

Until now, we have studied how to identify patterns in data. While some patterns are indisputably clear, in many cases we must figure out ways to distinguish systematic patterns from noise. Noise, also known as random error, is the irrelevant variation that occurs in every real-world data set. Quantifying the degree of statistical uncertainty of empirical findings is the topic for the *next* chapter, but this requires an understanding of probability. Probability is a set of mathematical tools that measure and model randomness in the world. As such, this chapter introduces the derivation of the fundamental rules of probability, with the use of mathematical notation. In the social sciences, we use probability to model the randomly determined nature of various real-world events, and even human behavior and beliefs. Randomness does not necessarily imply complete unpredictability. Rather, our task is to identify systematic patterns from noisy data.

### 5.1 Probability

We use *probability* as a measure of uncertainty. Probability is based on a set of three simple axioms, from which a countless number of useful theorems have been derived. In this section, we show how to define, interpret, and compute probability.

#### 5.1.1 FREQUENTIST VERSUS BAYESIAN

In everyday life, we often hear statements such as “the probability of winning a coin toss is 50%” and “the probability of Obama winning the 2008 US presidential election is 80%.” What do we mean by “probability”? There are at least two different interpretations. One interpretation, which is called the *frequentist* interpretation, states that probability represents the *limit* of relative frequency, defined as the ratio between the number of times the event occurs and the number of trials, in repeated trials under the same conditions. For example, the above statement about coin tosses can be interpreted as follows: if a coin toss is repeatedly conducted under the same conditions, the fraction of times a coin lands on heads approaches 0.5 as the number of coin tosses increases. Here, the mathematical term, “limit,” represents the

Figure 5.1. Reverend Thomas Bayes (1701–1761).  
Source: Wikipedia

value to which a sequence of relative frequencies converges as the number of (hypothetically) repeated experiments approaches infinity.

The frequentist interpretation of probability faces several difficulties. First, it is unclear what we mean by “the same conditions.” In the case of coin flips, such conditions may include initial angle and velocity as well as air pressure and temperature. However, if all conditions are identical, then the laws of physics imply that a coin flip will always yield the same outcome. Second, in practice, we can never conduct experiments like coin flips under the exact same conditions infinitely many times. This means that probability may be unable to describe the randomness of many events in the real world. In fact, coin flips may be among the easiest experiments to repeat under nearly identical conditions. Many other events covered in this book happen in dynamic social environments that are constantly changing.

How should we think about the probability of Obama winning the 2008 US presidential election from the frequentist perspective? Since the 2008 US presidential election occurs only once, it is strange to consider a hypothetical scenario in which this particular election occurs repeatedly under the same conditions. In addition, since Obama either wins the election or not, the probability of his victory should be either 0 or 1. Here, what is random is the election forecast (due to sampling variability, etc.), not the actual election outcome.

An alternative framework is the *Bayesian* interpretation of probability, named after an 18th century English mathematician and minister, Thomas Bayes (see figure 5.1). According to this paradigm, probability is a measure of one’s subjective belief about the likelihood of an event occurring. A probability of 0 means that an individual thinks an event is impossible, whereas a probability of 1 implies that the individual thinks the event is sure to happen. Any probability value between 0 and 1 indicates the degree to which one feels uncertain about the occurrence of the event. In contrast to the frequentist perspective, the Bayesian framework makes it easy to interpret the statement, “the probability of Obama winning the 2008 US presidential election is  $x\%$ ,” because  $x$  simply reflects the speaker’s subjective belief about the likelihood of Obama’s victory.

Critics of Bayesian interpretation argue that if scientists have identical sets of empirical evidence, they should arrive at the same conclusion rather than reporting different probabilities of the same event. Such subjectivity may hinder scientific progress because under the Bayesian framework, probability simply becomes a tool to describe one’s belief system.

In contrast, Bayesians contend that human beings, including scientists, are inherently subjective, so they should explicitly recognize the role of their subjective beliefs in scientific research.

Regardless of the ongoing controversy about its interpretation, probability was established as a mathematical theory by Soviet mathematician Andrey Kolmogorov in the early 20th century. Since both frequentists and Bayesians use this mathematical theory, the disagreement is about interpretation and is not mathematical.

There are two dominant ways to interpret probability. According to the **frequentist framework**, probability represents the limit of the relative frequency with which an event of interest occurs when the number of experiments repeatedly conducted under the same conditions approaches infinity. The **Bayesian framework**, in contrast, interprets probability as one's subjective belief about the likelihood of event occurrence.

### 5.1.2 DEFINITION AND AXIOMS

We define probability using the following three concepts: *experiment*, *sample space*, and *event*.

The definition of probability requires the following concepts:

1. **experiment**: an action or a set of actions that produce stochastic events of interest
2. **sample space**: a set of all possible outcomes of the experiment, typically denoted by  $\Omega$
3. **event**: a subset of the sample space

We can briefly illustrate each concept using the aforementioned two examples. Flipping a coin or holding an election would be the experiment, while the sample space would be given by  $\{\text{lands on heads}\}$ ,  $\{\text{lands on tails}\}$  or  $\{\text{Obama wins}\}$ ,  $\{\text{McCain wins}\}$ ,  $\{\text{a third-party candidate wins}\}$ . The mathematical term *set* refers to a collection of distinct objects. An event represents *any* subset of sample space, and hence it may include multiple outcomes. In fact, the entire sample space that contains all outcomes is also an event. Moreover, an event is said to *occur* if the set that defines the event includes an actual outcome of the experiment. In the election example, events include  $\{\text{Obama wins}\}$ ,  $\{\text{McCain wins}\}$ , which contains two outcomes and can be understood in English as “either Obama or McCain wins.” Since Obama won the election, this event did occur in 2008.

As another example, consider a voter’s decision in the 2008 US presidential election as an experiment. The idea is that a voter’s decision can be modeled as a stochastic, rather than deterministic, event. By considering all four possible outcomes, we can define the sample space of this experiment as  $\Omega = \{\text{abstain}\}, \{\text{vote for Obama}\}, \{\text{vote for McCain}\}, \{\text{vote for a third-party candidate}\}$ . Within this sample space, we may consider the occurrence of various events

including {vote for Obama}, {vote for McCain}, {vote for a third-party candidate} (i.e., “do not abstain”) and {abstain}, {vote for McCain},{vote for a third-party candidate} (i.e., “do not vote for Obama”).

We now discuss how to compute probability, starting with the simplest case in which all outcomes are equally likely to occur. In this case, the probability of event  $A$  occurring, denoted by  $P(A)$ , can be computed as the ratio of the number of elements in the corresponding set  $A$  to that in the entire sample space  $\Omega$ :

$$P(A) = \frac{\text{number of elements in } A}{\text{number of elements in } \Omega}. \quad (5.1)$$

To illustrate this, consider an experiment of tossing a fair coin three times. In this experiment, if we denote {lands on heads} and {lands on tails} as  $H$  and  $T$ , respectively, then the sample space is equal to the set of eight outcomes,  $\Omega = \{\text{HHH}\}, \{\text{HHT}\}, \{\text{HTH}\}, \{\text{HTT}\}, \{\text{THH}\}, \{\text{THT}\}, \{\text{TTH}\}, \{\text{TTT}\}$ . We can then compute the probability of, for example, landing on heads at least twice by counting the number of elements in the relevant set,  $A = \{\text{HHH}\}, \{\text{HHT}\}, \{\text{HTH}\}, \{\text{THH}\}$ . In this case, therefore, using equation (5.1) we obtain  $P(A) = 4/8 = 0.5$ .

Having defined probability, we next consider its basic rules or *axioms*. Modern probability theory rests on the following three simple axioms. Remarkably, from these axioms, the entire theory of probability, including all the existing rules and theorems, can be derived.

The **probability axioms** are given by the following three rules:

1. The probability of any event  $A$  is nonnegative:

$$P(A) \geq 0.$$

2. The probability that one of the outcomes in the sample space occurs is 1:

$$P(\Omega) = 1.$$

3. (*Addition rule*) If events  $A$  and  $B$  are mutually exclusive, then

$$P(A \text{ or } B) = P(A) + P(B). \quad (5.2)$$

The first two axioms together imply that probability ranges from 0 to 1. To understand the last axiom, recall the previous example in which the 2008 US presidential election is considered as an experiment. “Mutually exclusive” in the last axiom means that two events,  $A$  and  $B$ , do not share an outcome. As illustrated by the *Venn diagram* (named after John Venn, an English philosopher) in figure 5.2a, mutually exclusive events imply two disjoint sets, meaning that they do not share any element. Consider two events:  $A = \text{Obama wins}$  and  $B = \text{McCain wins}$ . Clearly, these two events are mutually exclusive in that both Obama and McCain cannot win at the same time. Hence, we can apply the addition rule to conclude that  $P(\{\text{Obama wins}\} \text{ or } \{\text{McCain wins}\}) = P(\text{Obama wins}) + P(\text{McCain wins})$ .

Now, consider two events that are not mutually exclusive because they share an outcome:  $A = \text{Obama loses}$  and  $B = \text{McCain loses}$ . In this case, the addition rule does not apply because both  $A$  and  $B$  contain the same outcome: a third-party candidate wins. For events that are not mutually exclusive, we can apply the following general addition rule.

For any given events  $A$  and  $B$ , the **addition rule** is given by

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B). \quad (5.3)$$

Applying this to the current example, we have  $P(\{\text{Obama loses}\} \text{ or } \{\text{McCain loses}\}) = P(\text{Obama loses}) + P(\text{McCain loses}) - P(\{\text{Obama loses}\} \text{ and } \{\text{McCain loses}\})$ .

This result can be immediately seen from the *Venn diagram* shown in figure 5.2b. In the diagram, we observe that the event,  $\{A \text{ or } B\}$ , can be decomposed into three mutually exclusive events,  $\{A \text{ and } B^c\}$  (gray region),  $\{B \text{ and } A^c\}$  (blue region), and  $\{A \text{ and } B\}$  (overlapped patterned region). The superscript  $c$  represents the *complement* of a set, which consists of all elements in the sample space except those in the set. For example,  $A^c$  represents the collection of all outcomes in the sample space that do not belong to  $A$ . The notation  $\{A \text{ and } B^c\}$  translates to “all outcomes of  $A$  that do not belong to  $B$ .” Since any outcome in the sample space belongs either to  $A$  or  $A^c$ , in general, we have

$$P(A^c) = 1 - P(A). \quad (5.4)$$

The equation directly follows from the probability axioms since events  $A$  and  $A^c$  are mutually exclusive and together they constitute the entire sample space.

Using the third probability axiom, given in equation (5.2), we have

$$P(A \text{ or } B) = P(A \text{ and } B^c) + P(B \text{ and } A^c) + P(A \text{ and } B). \quad (5.5)$$

When  $A$  and  $B$  are mutually exclusive,  $P(A \text{ and } B^c)$  and  $P(B \text{ and } A^c)$  reduce to  $P(A)$  and  $P(B)$ , respectively (see figure 5.2a). In addition, we have  $P(A \text{ and } B) = 0$  in this mutually exclusive case.

Finally, notice that event  $A$  can be decomposed as two mutually exclusive events,  $\{A \text{ and } B\}$  (overlapped patterned region) and  $\{A \text{ and } B^c\}$  (nonoverlapped gray region). This is called the *law of total probability*.

For any given events  $A$  and  $B$ , the **law of total probability** is given by

$$P(A) = P(A \text{ and } B) + P(A \text{ and } B^c). \quad (5.6)$$

According to the law of total probability, we can write  $P(A \text{ and } B^c) = P(A) - P(A \text{ and } B)$  by subtracting  $P(A \text{ and } B)$  from both sides of equation (5.6). Similarly, the law of total probability can be applied to event  $B$ , yielding  $P(B \text{ and } A^c) = P(B) - P(A \text{ and } B)$ . Substituting these results into equation (5.5) and simplifying the expression leads to the general addition rule given in equation (5.3). We emphasize that this result is obtained by using the

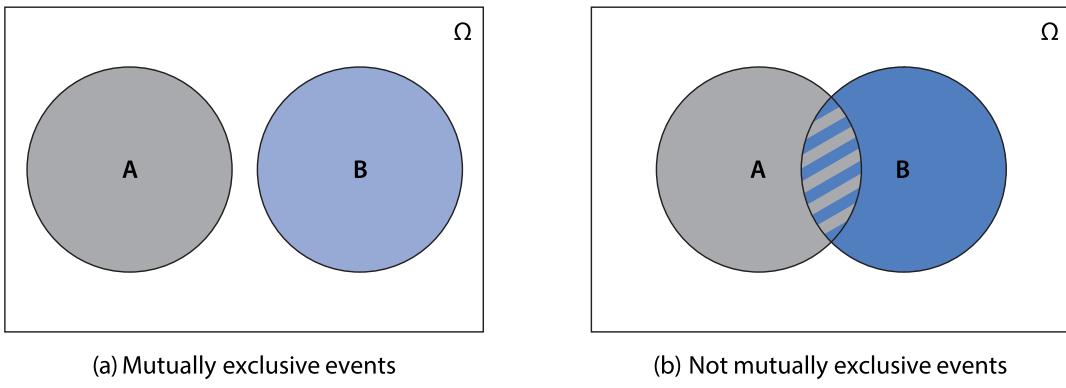


Figure 5.2. Venn Diagram. Two events,  $A$  and  $B$ , can be mutually exclusive, having two disjoint sets of outcomes (left plot) or not mutually exclusive, sharing some outcomes (right plot). The rectangle box represents the sample space  $\Omega$ .

probability axioms alone. In addition, readers are encouraged to confirm the results shown in equations (5.3)–(5.6) using the Venn diagram of figure 5.2.

### 5.1.3 PERMUTATIONS

When each outcome is equally likely, in order to compute the probability of event  $A$ , we need to count the number of elements in event  $A$  as well as the total number of elements in the sample space  $\Omega$  (see equation (5.1)). We introduce a useful counting technique, called *permutations*. Permutations refer to the number of ways in which objects can be arranged. For example, consider three unique objects  $A$ ,  $B$ , and  $C$ . There are six unique ways to arrange them:  $\{ABC, ACB, BAC, BCA, CAB, CBA\}$ .

How can we compute the number of permutations without enumerating every arrangement, especially when the number of objects is large? It turns out that there is an easy way to do this. Let's consider the above example of arranging three objects,  $A$ ,  $B$ , and  $C$ . First, there are three ways to choose the first object:  $A$ ,  $B$ , or  $C$ . Once the first object is selected, there are two ways to choose the second object. Finally, the third object remains, leaving us only one way to choose this last object. We can conceptualize this process as a tree shown in figure 5.3, where the total number of leaves equals the number of permutations. Thus, to compute the number of leaves, we only need to sequentially multiply the number of branches at each level, i.e.,  $3 \times 2 \times 1$ .

Generalizing this idea, we can compute the number of permutations of  $k$  objects out of a set of  $n$  unique objects, denoted by  ${}_nP_k$ , where  $k \leq n$ , using the following formula.

The number of **permutations** when arranging  $k$  objects out of  $n$  unique objects is given by

$${}_nP_k = n \times (n - 1) \times \cdots \times (n - k + 2) \times (n - k + 1) = \frac{n!}{(n - k)!}. \quad (5.7)$$

In this equation,  $!$  represents the *factorial* operator. When  $n$  is a nonnegative integer,  $n! = n \times (n - 1) \times \cdots \times 2 \times 1$ . Note that  $0!$  is defined as 1.

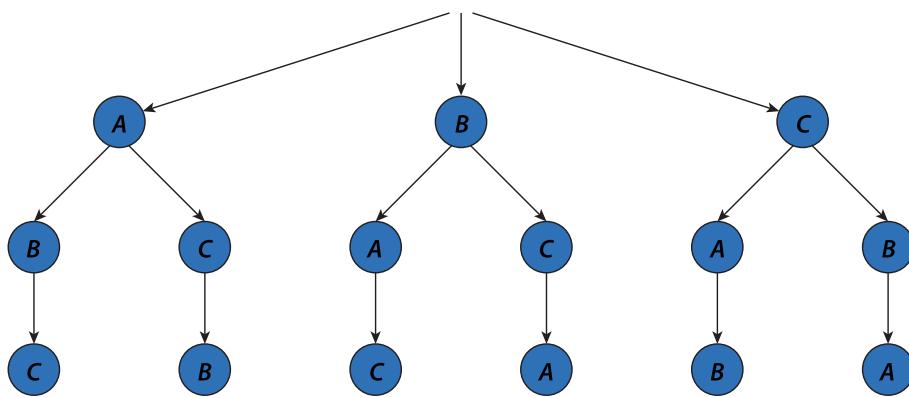


Figure 5.3. A Tree Diagram for Permutations. There are a total of six ways to arrange three unique objects.

In the previous example,  $n = 3$  and  $k = 3$ . Therefore,

$${}_3P_3 = \frac{3!}{0!} = \frac{3 \times 2 \times 1}{1} = 6.$$

As another example, compute the number of ways in which you can arrange 4 cards out of 13 unique cards. This can be computed by setting  $n = 13$  and  $k = 4$  in equation (5.7):

$${}_{13}P_4 = \frac{13!}{(13-4)!} = 13 \times 12 \times 11 \times 10 = 17160.$$

The *birthday problem* is a well-known counterintuitive example of permutations. The problem asks how many people one needs in order for the probability that at least two people have the same birthday to exceed 0.5, assuming that each birthday is equally likely. What is surprising about this problem is that the answer is only 23 people, which is much lower than what most people guess. To solve this problem using permutations, first notice the following relationship:

$$\begin{aligned} & P(\text{at least two people have the same birthday}) \\ &= 1 - P(\text{nobody has the same birthday}). \end{aligned} \tag{5.8}$$

This equality holds because the event {nobody has the same birthday} is the complement of the event {at least two people have the same birthday} (see equation (5.4)). This means that we only need to compute the probability that nobody has the same birthday.

Let  $k$  be the number of people. To compute the probability that nobody has the same birthday, we count the number of ways in which  $k$  people can have different birthdays. Since each birthday is assumed to be equally likely, we can use permutations to count the number of ways in which  $k$  unique birthdays can be arranged out of 365 days. This is given by  ${}_{365}P_k = 365!/(365-k)!$ . Applying equation (5.1), we then divide this number by the total number of elements in the sample space. The latter is equal to the total number of ways to select  $k$  possibly nonunique birthdays out of 365 days. The first person could have any of 365 days as his or her birthday, and so could any other person. Hence, the denominator is equal

to  $365 \times 365 \times \dots \times 365 = 365^k$ . Therefore, we have

$$\begin{aligned}
 & P(\text{nobody has the same birthday}) \\
 &= \frac{\# \text{ of ways in which } k \text{ unique birthdays can be arranged}}{\# \text{ of ways in which } k \text{ possibly nonunique birthdays can be arranged}} \\
 &= \frac{365P_k}{365^k} = \frac{365!}{365^k(365-k)!}. \tag{5.9}
 \end{aligned}$$

Together with equation (5.8), the solution to the birthday problem is  $1 - 365!/\{365^k(365-k)!\}$ .

Computing equation (5.9) is not easy even for a moderate value of  $k$  because both the denominator and numerator can take extremely large values. In such cases, it is often convenient to use the natural *logarithmic transformation* (see section 3.4.1). For the natural logarithm,  $e^A = B$  implies  $A = \log B$ . In addition, the basic rules of logarithms we use here are

$$\log AB = \log A + \log B, \quad \log \frac{A}{B} = \log A - \log B, \quad \text{and} \quad \log A^B = B \log A.$$

Applying these rules, we have

$$\log P(\text{nobody has the same birthday}) = \log 365! - k \log 365 - \log(365 - k)!.$$

After computing this probability on a logarithmic scale, we then take the exponential transformation of it to obtain the desired probability. In Stata, we use the `lnfactorial()` function to compute the logarithm of a factorial. To get the nonlogged value, we can wrap the `exp()` function around the logged value. We compute the probability that at least two people have the same birthday given  $k$ . The code is written so that it calculates values between a given 1 and 50 people. We first use `set obs` to create a data set with 50 observations. We then use `generate` with the built-in `_n` notation to sequentially number each observation, starting with 1. After calculating the log denominator, log numerator, and probability, we plot the results.

```

. * create data set with 50 observations
. clear
. set obs 50
number of observations (_N) was 0, now 50
. generate k = _n
.
. * calculate the log denominator and numerator
. generate logdenom = k * log(365) + lnfactorial(365 - k)
. generate lognumer = lnfactorial(365)
.

```

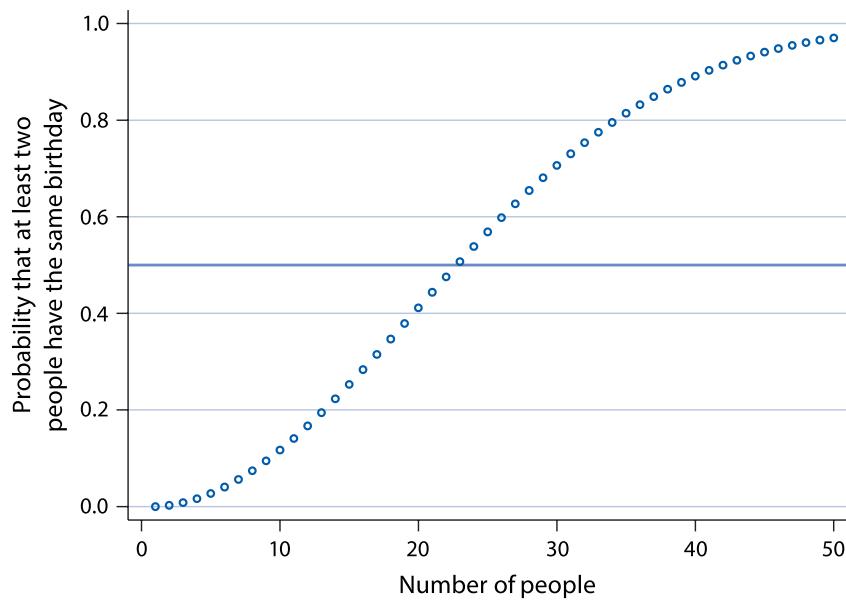
```

. * probability that at least two have the same birthday
. generate pr = 1 - exp(lognumer - logdenom)

.

. * plot probability
. scatter pr k, yline(.5) xtitle("Number of people") ///
>           ytitle("Probability that at least two" ///
>           "people have the same birthday") msymbol(oh)

```



We observe that when the number of people equals 23, the probability of at least two people having the same birthday exceeds 0.5. When the number of people is more than 50, this probability is close to 1.

#### 5.1.4 SAMPLING WITH AND WITHOUT REPLACEMENT

While we derived an exact analytical solution to the preceding birthday problem, we can also produce an approximate solution using a *Monte Carlo simulation method*. The name originates from the Monte Carlo Casino in Monaco, but we may also simply call it a *simulation* method. The Monte Carlo simulation method refers to a general class of stochastic (as opposed to deterministic) methods that can be used to approximately solve analytical problems by randomly generating quantities of interest.

For the birthday problem, we sample  $k$  possibly nonunique birthdays out of 365 days and check whether or not the sampled  $k$  birthdays are all different. We use *sampling with replacement* because for each of  $k$  draws, every one of 365 days is equally likely to be sampled *regardless of* which dates were sampled before. In other words, the fact that one person is born on a certain day of the year should not exclude someone else from being born on the same day. After repeating this sampling procedure many times, we compute the fraction

of simulation trials where at least two birthdays are the same, and this fraction serves as an estimate of the corresponding probability. This simulation procedure is intuitive because it emulates the *data-generating process*, or the actual process in which the data are generated, as described in the birthday problem.

We begin our simulation by creating an empty data set with 365 observations, one for each day. We generate a day number using the built-in indexing variable `_n`. Because our computation will involve random sampling, we also set a seed so our results can be replicated.

```
. clear
. * create data set and number observations
. set obs 365
number of observations (_N) was 0, now 365
. generate day = _n
. * set seed to ensure replicability
. set seed 12345
```

Prior to creating our loop, we have to indicate our sample size, the number of simulations we want, and create a counter that records each time people share the same birthday. The code that follows showcases some programming nuances in Stata. In section 4.1.1, we reviewed differences between scalars and macros in Stata. One difference is that scalars can be more efficient at holding single numeric values. Here, however, we use local macros to specify our sample size (`k`), number of simulations (`sims`), and events counter (`event`)—all single values.

```
. * specify the number of people to sample and simulations
. local k = 23
. local sims = 1000
. * create event counter
. local event = 0
```

The reason why we use local macros is because, unlike some commands such as `generate`, our loop command `forvalues` is a *nonevaluative* command. It is expecting a number but if we use a scalar, it will simply see the scalar name, generating an error. It will not process the scalar's content (i.e., number) unless forced. In contrast, we can request evaluation when first defining a local by using an equals sign. We could also define `k` and `sims` as variables using the `generate` command, but it is unnecessary to create new columns in our data set for these single values. In addition, defining `event` as a local prevents its value from resetting to zero with each restore and resample of our original data set.

To implement sampling with replacement in Stata, we use the `bsample` command. We would use the `sample` command for sampling without replacement. While unused in the birthday problem, *sampling without replacement* means that once an element is sampled, it will not be available for subsequent draws. For example, in the discussion of sample surveys in section 3.4.1, we introduced *simple random sampling* (SRS) as a method to randomly choose a

representative sample of respondents from a population. SRS is an example of sampling without replacement because we typically do not interview the same individual multiple times. For sampling with replacement, the basic syntax is `bsample k`, where `k` is the number of elements to choose, which we assigned as 23. We could feed a weights variable into the argument by using the `weight (varname)` option if we wanted to apply unequal probabilities in sampling each element. We could also ask Stata to sample by `strata()` or `cluster()`.

Having specified our sample size and number of observations, we can now begin our loop. We first draw our sample of `k` days, then we count the number of unique days drawn. If the number of unique days is less than `k`, that means there was more than one observation for at least one of the days (i.e., two or more people shared a birthday). By counting the number of rows stored in `r(r)`, generated by the `tabulate` command, we can identify the number of unique birthdays from each draw.

```
. * loop through simulations
. forvalues i = 1 / `sims' {
2.         preserve
3.                 * draw k samples
.                         bsample `k'
4.                 * count unique values drawn, saved in r(r)
.                         quietly tabulate day
5.                 * count if event occurred using if command
.                         * if there are duplicates, the number of unique birthdays
.                         * will be less than the number of birthdays, which is k
.                         if r(r) < `k' {
6.                             local event = `event' + 1
7.                         }
8.         restore
9. }
```

We add to our event counter each time the same day is drawn more than once using the `if` command. We have previously relied on the `if` qualifier for tasks such as subsetting the data. Here, we use the `if` command to change the value of our event counter. Stata will process the code within the curly braces that follow the `if` command only if the condition that follows `if` is met—in this case, if `r(r)` is less than `k`. We calculate the probability of two people sharing the same birthday in our simulation by dividing the number of times our event occurred by the total number of simulations.

```
. * fraction of trials where at least two birthdays are the same
. display `event' / `sims'
.522
```

While our simulation estimate is close to the analytical solution, which is .507, there is a small difference. This difference is called the *Monte Carlo error*, but is the inevitable consequence of the simulation approach. The size of the Monte Carlo error depends on the nature of the problem and it differs from one simulation to another. It is difficult to eliminate such an error

but it is possible to reduce it. To obtain a more accurate estimate, we run the same code with the number of simulations set to 100,000. Increasing the number of simulations gives us an estimate of .509, which is even closer to the true answer.

**Monte Carlo simulation method** refers to a general class of repeated random sampling procedures used to approximately solve analytical problems. Commonly used procedures include **sampling with replacement**, in which the same unit can be repeatedly sampled, and **sampling without replacement**, in which each unit can be sampled at most once.

### 5.1.5 COMBINATIONS

We introduce another useful counting method called *combinations*. Combinations are similar to permutations, but the former ignores ordering while the latter does not. That is, combinations are ways to choose  $k$  distinct elements out of  $n$  elements without regard to their order. This means that when choosing two elements, two *different* permutations,  $AB$  and  $BA$ , represent one *identical* combination. Since the order in which the elements are arranged does not matter, the number of combinations is never greater than the number of permutations. For example, if we choose two distinct elements out of three elements,  $A$ ,  $B$ , and  $C$ , the number of permutations is  ${}_3P_2 = 6$  ( $AB$ ,  $BA$ ,  $AC$ ,  $CA$ ,  $BC$ ,  $CB$ ), whereas the number of combinations is three ( $AB$ ,  $AC$ ,  $BC$ ).

In fact, to compute combinations, we first calculate permutations  ${}_nP_k$  and then divide by  $k!$ . This is because given  $k$  sampled elements, there are  $k!$  ways to arrange them in a different order, and yet all these arrangements are counted as a single combination. In the preceding example, for every set of two sampled elements (e.g.,  $A$  and  $B$ ), we have  $2!$  ( $= 2 \times 1 = 2$ ) ways of arranging them (i.e.,  $AB$  and  $BA$ ) but these two permutations count as one combination. Here, we obtain the number of combinations through the division of  ${}_3P_2$  by  $2!$ . In general, the formula for combinations is given as follows.

The number of **combinations** when choosing  $k$  distinct elements from  $n$  elements is denoted by either  ${}_nC_k$  or  $\binom{n}{k}$  and is computed as

$${}_nC_k = \binom{n}{k} = \frac{{}_nP_k}{k!} = \frac{n!}{k!(n-k)!}. \quad (5.10)$$

Suppose that we are creating a committee of 5 out of 20 people (10 men and 10 women). Assume that each person is equally likely to be assigned to the committee. What is the probability that at least 2 women are on the committee? To compute this probability, we first note the following equality:

$$P(\text{at least 2 women are on the committee})$$

$$= 1 - P(\text{no woman is on the committee}) - P(\text{exactly 1 woman is on the committee}).$$

To the Members of the California State Assembly:

I am returning Assembly Bill 1176 without my signature.

For some time now I have lamented the fact that major issues are overlooked while many unnecessary bills come to me for consideration. Water reform, prison reform, and health care are major issues my Administration has brought to the table, but the Legislature just kicks the can down the alley.

Yet another legislative year has come and gone without the major reforms Californians overwhelmingly deserve. In light of this, and after careful consideration, believe it is unnecessary to sign this measure at this time.

Sincerely,

Arnold Schwarzenegger

Figure 5.4. California Governor Arnold Schwarzenegger's Veto Message in 2009.

To compute the two probabilities on the right-hand side of this equation, we count the total number of ways we can assign 5 people to the committee out of 20 people regardless of their gender. This is given by  ${}_{20}C_5 = 15,504$ . The number of ways in which we can have no woman on the committee is given by  ${}_{10}C_0 \times {}_{10}C_5 = 252$  because there is  ${}_{10}C_0$  way to choose no woman and there are  ${}_{10}C_5$  ways to choose 5 out of 10 men. Thus, the probability of having no woman is 0.016. The number of ways in which we can have exactly 1 woman on the committee is  ${}_{10}C_1 \times {}_{10}C_4 = 2,100$ , giving a probability of 0.135. Altogether, the probability of having at least 2 women on the committee equals  $0.84 = 1 - 0.16 - 0.135$ .

As a more complex example of combinations, we discuss an incident that occurred in 2009 when California Governor Arnold Schwarzenegger wrote a message to the state assembly regarding his veto of Assembly Bill 1176.<sup>1</sup> This message is displayed in figure 5.4. When the message was released, many observed that the first letters of each line in the main text, starting with "F" and ending with "u," constitute a sentence of profanity. Asked whether this was intentional, Schwarzenegger's spokesman replied, "My goodness. What a coincidence. I suppose when you do so many vetoes, something like this is bound to happen." We consider the probability of this acrostic happening by chance.

For the sake of simplicity, suppose that the governor gave his veto message to his secretary, who then typed it in her computer but hit the return key at random. That is, the 85 words ("For" to "time") were divided by (random) line breaks into 7 lines, each with at least one word. We further assume that there are no broken words, every way of breaking the lines was equally likely, and the total number of lines is fixed at seven. Under this scenario, what is the probability of the coincidence happening?

To compute this probability using equation (5.1), we first consider the number of ways in which the 85 words can be divided into 7 lines. Note that to end up with 7 lines, 6 line

<sup>1</sup>This section is based on Philip B. Stark (2009). "Null and vetoed: Chance coincidence?" *Chance*, vol. 23, no. 4, pp. 43–46. Although there are a total of 86 words from "For" to "time," we follow the original article and use 85.

breaks must be inserted. A line break may be inserted before the second word, before the third word, ..., or before the 85th word. There are thus 84 places into which 6 line breaks must be inserted. How many ways can we insert line breaks into 6 out of these 84 places? To compute this number, we use combinations rather than permutations because the order in which 6 line breaks are inserted does not matter. (Of course, the words in the acrostic must be ordered in a particular way to generate the profanity.) Therefore, the application of combinations leads to  ${}_{84}C_6 = 84!/(6!78!)$  equally likely partitions. To compute combinations in Stata, we use the `comb()` function. We add `%12.0f` to the `display` command to see the full number. Stata otherwise displays very large and very small numbers in exponential format.

```
. display %12.0f comb(84, 6)
406481544
```

There are more than 400 million ways to insert 6 line breaks. However, there are only 12 ways to produce this particular acrostic. The break to produce “u” at the beginning of the second line can be in only one place (“unnecessary”). The break to produce “c” at the beginning of the third line can happen in any of three places (“come,” “consideration,” “care”). The break for the “k” can be in only one place (“kicks”). The break for the “y” can be in any of two places (“Yet,” “year”). The break for the “o” can be in any of two places (“overwhelmingly,” “of”). The break for the second “u” can be in only one place (“unnecessary”). These scenarios lead to  $12 = 1 \times 3 \times 1 \times 2 \times 2 \times 1$ . Hence, the probability that this randomization scheme would produce the acrostic is  $12/{}_{84}C_6$ , or about 1 in 34 million. The analysis suggests that according to this probabilistic model, the “coincidence” is a highly unlikely event.

## 5.2 Conditional Probability

We next introduce conditional probability, which concerns how the probability of an event changes after we observe other events. Conditional probability follows the rules of probability described in section 5.1. The difference is that conditional probability enables us to take into account observed evidence.

### 5.2.1 CONDITIONAL, MARGINAL, AND JOINT PROBABILITIES

We begin by defining the conditional probability of event  $A$  occurring, given the information that event  $B$  has occurred. This conditional probability, denoted as  $P(A | B)$ , has the following definition.

The **conditional probability** of event  $A$  occurring given that event  $B$  occurred is defined as

$$P(A | B) = \frac{P(A \text{ and } B)}{P(B)}. \quad (5.11)$$

In this equation,  $P(A \text{ and } B)$  is the **joint probability** of both events occurring, whereas  $P(B)$  is the **marginal probability** of event  $B$ . By rearranging, we obtain the **multiplication rule**,

$$P(A \text{ and } B) = P(A | B)P(B) = P(B | A)P(A). \quad (5.12)$$

Using this rule, we can derive an alternative form of the **law of total probability** introduced in equation (5.6),

$$P(A) = P(A | B)P(B) + P(A | B^c)P(B^c). \quad (5.13)$$

To see the importance of conditioning, consider two couples who are both expecting twins. One couple had an ultrasound exam, but the technician was able to determine only that one of the two was a boy. The other couple did not find out the genders of their twins until the delivery when they saw the first baby was a boy. What is the probability that both babies are boys? Is this probability different between the two couples? We begin by noting that there are four outcomes in the sample space. Denoting the baby gender by “G” for girl and “B” for boy, respectively, we can represent the sample space by  $\Omega = \{GG, GB, BG, BB\}$ . For example,  $GB$  means that the elder twin is a girl and the younger one is a boy.

Then, for the first couple, the probability of interest is

$$\begin{aligned} P(BB | \text{at least one is a boy}) &= \frac{P(BB \text{ and } \{\text{at least one is a boy}\})}{P(\text{at least one is a boy})} \\ &= \frac{P(BB \text{ and } \{BB \text{ or } BG \text{ or } GB\})}{P(BB \text{ or } BG \text{ or } GB)} \\ &= \frac{P(BB)}{P(BB \text{ or } BG \text{ or } GB)} = \frac{1/4}{3/4} = \frac{1}{3}. \end{aligned}$$

The third equality follows from the fact that event  $BB$  is a subset of event {at least one is a boy}, i.e.,  $BB$  and  $\{BB \text{ or } BG \text{ or } GB\} = BB$ .

In contrast, for the second couple, we have

$$\begin{aligned} P(BB | \text{elder twin is a boy}) &= \frac{P(BB \text{ and } \{\text{the elder twin is a boy}\})}{P(\text{elder twin is a boy})} \\ &= \frac{P(BB \text{ and } \{BB \text{ or } BG\})}{P(BB \text{ or } BG)} \\ &= \frac{P(BB)}{P(BB \text{ or } BG)} = \frac{1/4}{1/2} = \frac{1}{2}. \end{aligned}$$

This example illustrates that the information upon which we condition matters. Knowing that the first baby is a boy, as opposed to knowing that at least one is a boy, gives a different conditional probability of the same event.

**Table 5.1.** The Names and Descriptions of Variables in the Florida Registered Voter List Sample.

Variable	Description
surname	surname
county	county ID of voter's residence
vtd	voting district ID of the voter's residence
age	age
gender	gender: m = male and f = female
race	self-reported race

Probability and conditional probability can also be used to describe the characteristics of a population. For example, if 10% of a population of voters are black, then we may write  $P(\text{black}) = 0.1$ . We can interpret this probability as stating that if we randomly sample a voter from this population there is a 10% chance this voter is black. Similarly,  $P(\text{black} | \text{hispanic or black})$  represents the population proportion of blacks among minority (i.e., black and Hispanic) voters.

As an illustration, we will use a random sample of 10,000 registered voters from Florida contained in the Stata file `FLVoters.dta`. Table 5.1 shows the names and descriptions of variables in this sample list of registered voters. To begin, we load the data and remove those voters who contain a missing value on race, gender, or age using the missing function.

```

. cd probability
. use FLvoters, clear
. drop if missing(race, gender, age)
(887 observations deleted)

. describe, short
Contains data from FLvoters.dta
obs: 9,113
vars: 6
Sorted by:
Note: Dataset has changed since last saved.
21 Apr 2020 20:10

```

We can see that 887 observations were dropped from the data set and 9,113 remain. For the sake of illustration, we will treat this sample of 9,113 as a population of interest. To compute the *marginal probability* for each racial category, we can use the tabulate command to show the proportion of voters who belong to each racial group in this population.

```
. tabulate race
```

race	Freq.	Percent	Cum.
asian	175	1.92	1.92
black	1,194	13.10	15.02
hispanic	1,192	13.08	28.10
native	29	0.32	28.42
other	310	3.40	31.82
white	6,213	68.18	100.00
Total	9,113	100.00	

If we divide the results by 100 to convert the percent into a decimal value, we can see, for example, that  $P(\text{black}) = 0.13$  and  $P(\text{white}) = 0.68$ . Similarly, we can obtain the marginal probabilities of gender as follows.

```
. tabulate gender
```

gender	Freq.	Percent	Cum.
f	4,883	53.58	53.58
m	4,230	46.42	100.00
Total	9,113	100.00	

This yields  $P(\text{female}) = 0.54$  and  $P(\text{male}) = 0.46$ . Next, to compute the *conditional probability* of race given gender, we can look at the proportion of each racial group among female voters and among male voters, separately. We use `tabulate` and ask for the percentages by column. We also turn off the frequency counts with the `nofreq` option.

```
. tabulate race gender, column nofreq
```

race	gender		Total
	f	m	
asian	1.70	2.17	1.92
black	13.88	12.20	13.10
hispanic	13.64	12.43	13.08
native	0.35	0.28	0.32
other	3.24	3.59	3.40
white	67.19	69.31	68.18
Total	100.00	100.00	100.00

The result suggests, for example,  $P(\text{black} \mid \text{female}) = 0.14$  and  $P(\text{white} \mid \text{female}) = 0.67$ . Lastly, the *joint probability* of race and gender can be computed by calculating the proportion of voters who belong to specific racial and gender groups. To calculate this, we specify the `cell` option after the `tabulate` command, which provides each cell's relative frequency.

```
. tabulate race gender, cell norefq
```

race	gender		Total
	f	m	
asian	0.91	1.01	1.92
black	7.44	5.66	13.10
hispanic	7.31	5.77	13.08
native	0.19	0.13	0.32
other	1.73	1.67	3.40
white	36.00	32.17	68.18
Total	53.58	46.42	100.00

This joint probability table gives  $P(\text{black and female}) = 0.07$  and  $P(\text{white and male}) = 0.32$ . From this joint probability, we can compute the marginal and conditional probability. We place our results in table 5.2 for further clarity. First, to obtain the marginal probability, we apply the *law of total probability* given in equation (5.6). For example, we can compute the probability of being a black voter by

$$P(\text{black}) = P(\text{black and female}) + P(\text{black and male}).$$

Thus, summing over columns for each row results in the marginal probability of race. As shown in the last column of the last table, this operation yields results identical to the ones we obtained when running `tabulate race`.

Similarly, we can obtain the marginal probability of gender from the joint probability table by summing over racial categories. Since we have a total of six racial categories, we will extend the law of total probability given in equation (5.6) to the following:

$$P(A) = \sum_{i=1}^N P(A \text{ and } B_i), \quad (5.14)$$

where  $B_1, \dots, B_N$  is a set of mutually exclusive events that together cover the entire sample space. In the current setting, for example, since racial categories are mutually exclusive, we have

$$\begin{aligned} P(\text{female}) &= P(\text{female and asian}) + P(\text{female and black}) + P(\text{female and hispanic}) \\ &\quad + P(\text{female and native}) + P(\text{female and other}) + P(\text{female and white}). \end{aligned}$$

**Table 5.2.** An Example of a Joint Probability Table.

	Gender		
	Female	Male	Marg prob
Asian	0.91	1.01	1.92
Black	7.44	5.66	13.10
Hispanic	7.31	5.77	13.08
Native	0.19	0.13	0.32
Other	1.73	1.67	3.40
White	36.00	32.17	68.18
Marg prob	53.58	46.42	100.00

The table is based on Florida voter registration data. The marginal probability of gender (far right column) and that of race (bottom row) can be obtained by summing the joint probabilities over columns and over rows, respectively.

Therefore, the marginal probability of gender is obtained by summing over rows for each column of the joint probability table, as shown in the last row of table 5.2.

Finally, the *conditional probability* can be obtained as the ratio of joint probability to the marginal probability (see equation (5.11)). For example, the conditional probability of being black among female voters is calculated as

$$P(\text{black} \mid \text{female}) = \frac{P(\text{black and female})}{P(\text{female})} \approx \frac{0.074}{0.536} \approx 0.138,$$

which, as expected, is equal to what we computed earlier.

The results of this example are summarized in table 5.2. From the joint probability, both marginal and conditional probabilities can be obtained. To compute marginal probability, we sum over either rows or columns. Once marginal probability is obtained in this way, we can divide joint probability by marginal probability in order to calculate the desired conditional probability.

We can extend the definition of conditional probability to settings with more than two types of events. For events  $A$ ,  $B$ , and  $C$ , the joint probability is defined as  $P(A \text{ and } B \text{ and } C)$ , whereas there are three marginal probabilities:  $P(A)$ ,  $P(B)$ , and  $P(C)$ . In this case, there are two types of conditional probabilities: the joint probability of two events conditional on the remaining event (e.g.,  $P(A \text{ and } B \mid C)$ ) and the conditional probability of one event given the other two (e.g.,  $P(A \mid B \text{ and } C)$ ). These conditional probabilities can be defined analogously to the two-event case as

$$P(A \text{ and } B \mid C) = \frac{P(A \text{ and } B \text{ and } C)}{P(C)}, \quad (5.15)$$

$$P(A \mid B \text{ and } C) = \frac{P(A \text{ and } B \text{ and } C)}{P(B \text{ and } C)} = \frac{P(A \text{ and } B \mid C)}{P(B \mid C)}. \quad (5.16)$$

The second equality in equation (5.16) follows from the equality  $P(A \text{ and } B \text{ and } C) = P(A \text{ and } B | C)P(C)$ , which is obtained by rearranging the terms in equation (5.15).

To illustrate the above conditional probabilities, we create a new agegroup variable indicating four age groups: 20 and below, 21–40, 41–60, and above 60.

```
. generate agegroup = .
(9,113 missing values generated)

. replace agegroup = 1 if age <= 20
(161 real changes made)

. replace agegroup = 2 if age > 20 & age <= 40
(2,469 real changes made)

. replace agegroup = 3 if age > 40 & age <= 60
(3,285 real changes made)

. replace agegroup = 4 if age > 60 & age < .
(3,198 real changes made)
```

Stata can produce separate tables using the `by` option. For instance, we could tabulate age group and race by gender using `bysort gender: tabulate race agegroup`. That would create the joint probabilities within each gender, but not overall. Instead, we use `egen` with the `group()` function to create a composite variable that combines race and gender into one variable. We then create a two-way table with our age group variable.

group(gender race)	agegroup				Total
	1	2	3	4	
f asian	0.01	0.26	0.42	0.22	0.91
f black	0.16	2.81	2.58	1.89	7.44
f hispanic	0.15	2.60	2.73	1.82	7.31
f native	0.01	0.04	0.07	0.07	0.19
f other	0.03	0.63	0.58	0.49	1.73
f white	0.59	7.97	12.61	14.84	36.00
m asian	0.02	0.29	0.52	0.19	1.01
m black	0.16	2.28	1.90	1.32	5.66
m hispanic	0.16	1.98	2.22	1.42	5.77
m native	0.00	0.04	0.03	0.05	0.13
m other	0.04	0.69	0.56	0.37	1.67
m white	0.41	7.51	11.84	12.42	32.17
Total	1.77	27.09	36.05	35.09	100.00

The proportion of black female voters who are above 60 or  $P(\text{black and above 60 and female})$  is equal to 0.019. Suppose that we wish to obtain the conditional probability of being black and female given that a voter is above 60 years old or  $P(\text{black and female} | \text{above 60})$ . Using equation (5.15), we can compute this conditional probability by dividing the joint probability by the marginal probability of being above 60 or  $P(\text{above 60})$ . We use the `count` command to obtain the number (or frequency) of black females over the age of 60 in our sample. We also count the number of people over 60. We divide these counts by the total sample size, stored in `_N`, to show the joint probability of being a black female over 60 and the marginal probability of being over 60. To obtain the conditional probability, we then divide the frequency of being a black female by the frequency of being over age 60.

```

. * joint probability of black female over 60
. count if race == "black" & gender == "f" & agegroup == 4
172

. scalar blackfem60 = r(N)

. display blackfem60 / _N
.01887414

.

. * marginal probability of being over 60
. count if agegroup == 4
3,198

. scalar age60 = r(N)

. display age60 / _N
.35092725

.

. * P(black and female | above 60)
. display blackfem60 / age60
.05378361

```

According to equation (5.16), the conditional probability of being black given that a voter is female and above 60 years old or  $P(\text{black} | \text{female and 60 years old})$  can be computed by dividing the three-way joint probability  $P(\text{black and above 60 and female})$  by the two-way joint probability  $P(\text{above 60 and female})$ . To obtain this two-way joint probability, we can create a two-way joint probability table for age group and gender.

```

. * two-way joint probability table for age group and gender
. tabulate agegroup gender, cell nofreq

```

agegroup	gender		Total
	f	m	
1	0.97	0.80	1.77
2	14.31	12.78	27.09

3	18.98	17.06	36.05
4	19.32	15.77	35.09
Total	53.58	46.42	100.00

```

. count if agegroup == 4 & gender=="f"
1,761

. scalar fem60 = r(N)

.

. * P(above 60 and female)
. display fem60 / _N
.19324043

. * P(black | female and above 60)
. display blackfem60 / fem60
.09767178

```

### 5.2.2 INDEPENDENCE

Having defined conditional probability, we can now formally discuss the concept of *independence*. Intuitively, the independence of two events implies that the knowledge of one event does not give us any additional information about the occurrence of the other event. That is, if events  $A$  and  $B$  are independent of each other, the conditional probability of  $A$  given  $B$  does not differ from the marginal probability of  $A$ . Similarly, the conditional probability of  $B$  given  $A$  does not depend on  $A$ :

$$P(A | B) = P(A) \quad \text{and} \quad P(B | A) = P(B). \quad (5.17)$$

Together with equation (5.12), this equality implies the following formal definition of independence between events  $A$  and  $B$ .

Events  $A$  and  $B$  are **independent** if and only if the joint probability is equal to the product of the marginal probabilities:

$$P(A \text{ and } B) = P(A)P(B). \quad (5.18)$$

We investigate whether race and gender are independent of each other in the sample of Florida registered voters analyzed earlier. Although we do not expect this relationship to be exactly independent, we examine whether the proportion of female voters, for example, is greater than expected in some racial groups. Note that if independence holds, we should have  $P(\text{black and female}) = P(\text{black})P(\text{female})$ ,  $P(\text{white and male}) = P(\text{white})P(\text{male})$ , and so on. We compare the products of marginal probabilities for race and female with their joint probabilities using a scatterplot. For this analysis, we make use of another important programming concept in Stata: matrices.

So far, we have used scalars and local macros to store values and information. Matrices also offer a temporary yet more complex storage structure. Like data sets, *matrices* are spreadsheet-like, rectangular arrays comprised of rows and columns. However, while a data set can store both string and numeric variables, matrices store only numeric values. In addition, cells of a matrix are referred to by row and column, (with the syntax *matrixname* [*row*, *column*]). Matrices are used to store summary (r-class) and estimation (e-class) results. To demonstrate their utility, we rerun `tabulate` commands from the last section and store the results in a matrix. We do this with the aid of the user-written `estout` package. To install this package, type `net search estout`, click on the first link and install. The package's `estpost` command allows us to store the `tabulate` output as e-class results in a matrix. The package is useful for producing tables (see `help estout` for more), but as we will show, its functionality can be applied to other tasks, such as plotting results.

<code>. estpost tabulate race gender</code>				
gender	race	e (b)	e (pct)	e (colpct)
f				
	asian	83	.9107868	1.699775
	black	678	7.439921	13.88491
	hispanic	666	7.308241	13.63916
	native	17	.1865467	.3481466
	other	158	1.733787	3.235716
	white	3281	36.00351	67.1923
	Total	4883	53.58279	100
m				
	asian	92	1.009547	2.174941
	black	516	5.662241	12.19858
	hispanic	526	5.771974	12.43499
	native	12	.13168	.2836879
	other	152	1.667947	3.593381
	white	2932	32.17382	69.31442
	Total	4230	46.41721	100
Total				
	asian	175	1.920334	1.920334
	black	1194	13.10216	13.10216
	hispanic	1192	13.08022	13.08022
	native	29	.3182267	.3182267
	other	310	3.401734	3.401734
	white	6213	68.17733	68.17733
	Total	9113	100	100

The `estpost tabulate` command stores the frequency `e(b)`, the cell percentages `e(pct)`, the column percentages `e(colpct)`, and the row percentages `e(rowpct)` for each outcome, naming each column and row accordingly. We access and view the `e()` results using the `matrix` command with `list`.

```
. matrix list e(pct)

e(pct)[1,21]
f:          f:          f:          f:          f:          f:          f:          m:
asian       black      hispanic     native      other      white    Total   asian
c1 .91078679 7.439921 7.308241 .18654669 1.7337869 36.003511 53.582794 1.0095468

m:          m:          m:          m:          m:          m:          Total:  Total:
black      hispanic     native      other      white    Total   asian   black
c1 5.6622408 5.7719741 .13168002 1.6679469 32.173818 46.417206 1.9203336 13.102162

Total:      Total:      Total:      Total:      Total:      Total:
hispanic    native      other      white    Total
c1 13.080215 .31822671 3.4017338 68.177329      100
```

The joint probabilities of being female by race are in the first six columns of the first row of the `e(pct)` matrix. Summing these values gives us the marginal probability of being a female voter, stored in the seventh column. Using an apostrophe, we transpose the matrix so that the cell percentages are in rows and save it as a new matrix called `pct`. Transposing the matrix so values are stored in one column will allow us to convert the values from our matrices into variables that we can then plot. We extract and save values from the first six rows of the first (and here, only) column of the transposed matrix into a new matrix called `joint_fr` by specifying `pct[1..6,1]`, dividing the contents by 100 to convert the percent to a decimal value. We save the marginal probability of being female as a scalar called `margin_f` because it contains just a single value.

```
. * transpose cell percentages
. matrix pct = e(pct)'

. * show transposed matrix to demonstrate
. matrix list pct

pct[21,1]
c1
f:asian  .91078679
f:black   7.439921
f:hispanic 7.308241
f:native   .18654669
f:other    1.7337869
f:white    36.003511
f:Total    53.582794
m:asian   1.0095468
m:black   5.6622408
```

```

m:hispanic 5.7719741
m:native   .13168002
m:other    1.6679469
m:white    32.173818
m:Total    46.417206
Total:asian 1.9203336
Total:black 13.102162
Total:hispanic 13.080215
Total:native .31822671
Total:other  3.4017338
Total:white  68.177329
Total:Total   100
.
* joint probability of being female, by race
matrix joint_rf = pct[1..6,1] / 100
.
* marginal probability of being female
scalar margin_f = pct[7,1] / 100

```

We repeat these commands for the column percentages. The marginal probabilities for each race are in the six penultimate columns of the `e(colpct)` matrix (viewable using `matrix list e(colpct)`), which we again transpose to more easily convert into variables. We extract the six penultimate rows of the first column from the transposed matrix and divide the values by 100. We then calculate  $P(\text{race}) \times P(\text{female})$  and join the two matrices using the `,` operator.<sup>2</sup> The contents of the matrices will be stored as separate columns in the combined matrix.

```

.
* transpose column percentages
matrix colpct = e(colpct) '
.
* marginal probability of race
matrix margin_r = colpct[15..20,1] / 100
.
.
* calculate  $P(\text{race}) \times P(\text{female})$ 
matrix product_rf = margin_r * margin_f
.
* combine matrices
matrix joint_marg = joint_rf, product_rf

```

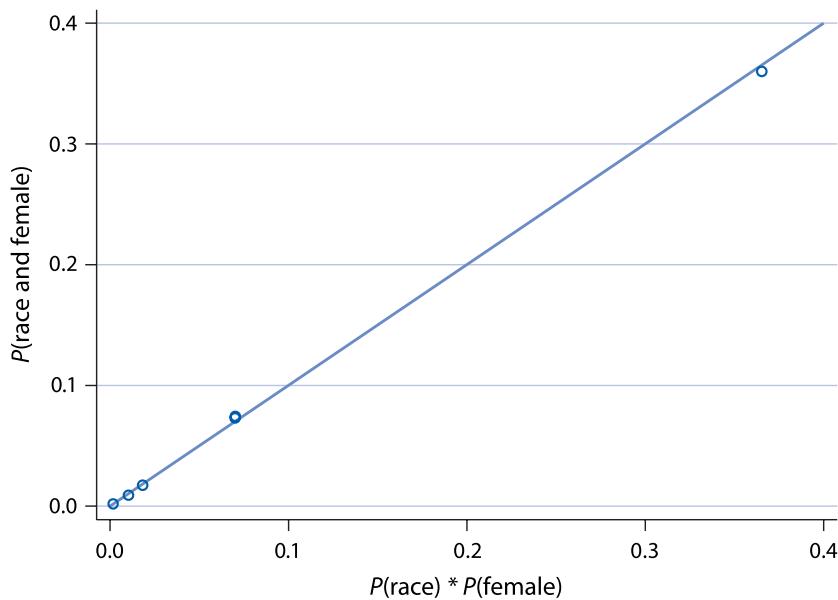
The `svmat` command allows us to store our matrix columns as variables, creating `joint_marg1` and `joint_marg2`. We can now plot our results. We include function `y = x, range()` in our plot command to add a 45-degree line. The scatterplot shows that the points fall neatly along the 45-degree line, implying that  $P(\text{race})P(\text{female})$  (horizontal axis) and  $P(\text{race and female})$  (vertical axis) are approximately equal. This means that race and gender are approximately independent in this sample of registered voters. That is, the

<sup>2</sup>If we wanted to add a matrix as rows, we would use double backslashes.

knowledge of a voter's gender does not help us predict her race. Similarly, one's race does not predict gender either.

```
. * save columns as variables
. svmat joint_marg

.
. * scatter plot
. scatter joint_marg1 joint_marg2, msymbol(Oh) || function y=x, range(0 .4) ///
>         xtitle("P(race) * P(female)") ytitle("P(race and female)") ///
>         legend(off)
```



The notion of independence extends to situations with more than two events. For example, if we have three events  $A$ ,  $B$ , and  $C$ , the *joint independence* among these events implies that the joint probability can be written as the product of marginal probabilities:

$$P(A \text{ and } B \text{ and } C) = P(A)P(B)P(C). \quad (5.19)$$

Furthermore, we can define the independence between two events conditional on another event. The *conditional independence* of events  $A$  and  $B$  given event  $C$  implies that the joint probability of  $A$  and  $B$  given  $C$  is equal to the product of two conditional probabilities:

$$P(A \text{ and } B | C) = P(A | C)P(B | C). \quad (5.20)$$

Joint independence given in equation (5.19) implies pairwise independence given in equation (5.18). This result can be obtained by applying the *law of total probability*:

$$\begin{aligned}
 P(A \text{ and } B) &= P(A \text{ and } B \text{ and } C) + P(A \text{ and } B \text{ and } C^c) \\
 &= P(A)P(B)P(C) + P(A)P(B)P(C^c) \\
 &= P(A)P(B)(P(C) + P(C^c)) = P(A)P(B).
 \end{aligned}$$

In addition, joint independence implies conditional independence, defined in equation (5.20), but the converse is not necessarily true. This result is based on the definition of conditional probability given in equation (5.15):

$$P(A \text{ and } B | C) = \frac{P(A \text{ and } B \text{ and } C)}{P(C)} = \frac{P(A)P(B)P(C)}{P(C)} = P(A | C)P(B | C).$$

The last equality follows from the fact that joint independence implies pairwise independence (and hence equation (5.17) holds for  $A$  and  $C$  as well as  $B$  and  $C$ ).

To examine joint independence among our sample of registered Florida voters, we compare the elements of the three-way proportion on page 216 with the corresponding product of marginal probabilities for race, age, and gender. We saved the marginal probabilities for race and for gender in the last example. We now have to save the marginal probabilities for age, the joint probability for age and gender, and the joint probabilities for age, race, and gender, using the same `estpost` command as before. As an illustration, we set the age group to the above 60 category and examine female voters. We also examine the conditional independence between race and gender given age. For this, we again set the age and gender groups to the above 60 and female categories, respectively. The results show that both joint (left-hand plot) and conditional (right-hand plot) independence relationships approximately hold, despite small deviations.

```

. * joint probability of being 60+ female
. quietly estpost tabulate agegroup gender

. matrix pct2 = e(pct) `

. scalar joint_f60 = pct2[4,1] / 100

.

. * marginal probability for 60+
. scalar margin_60 = pct2[14,1]

.

. * joint probability for 60+ female and race
. quietly estpost tabulate genderrace agegroup

. matrix pct3 = e(pct) `

. matrix joint_60rf = pct3[40..45,1] / 100

.

. * joint independence
. matrix j_product_60rf = (product_rf * margin_60) / 100

. matrix joint_ind = joint_60rf, j_product_60rf

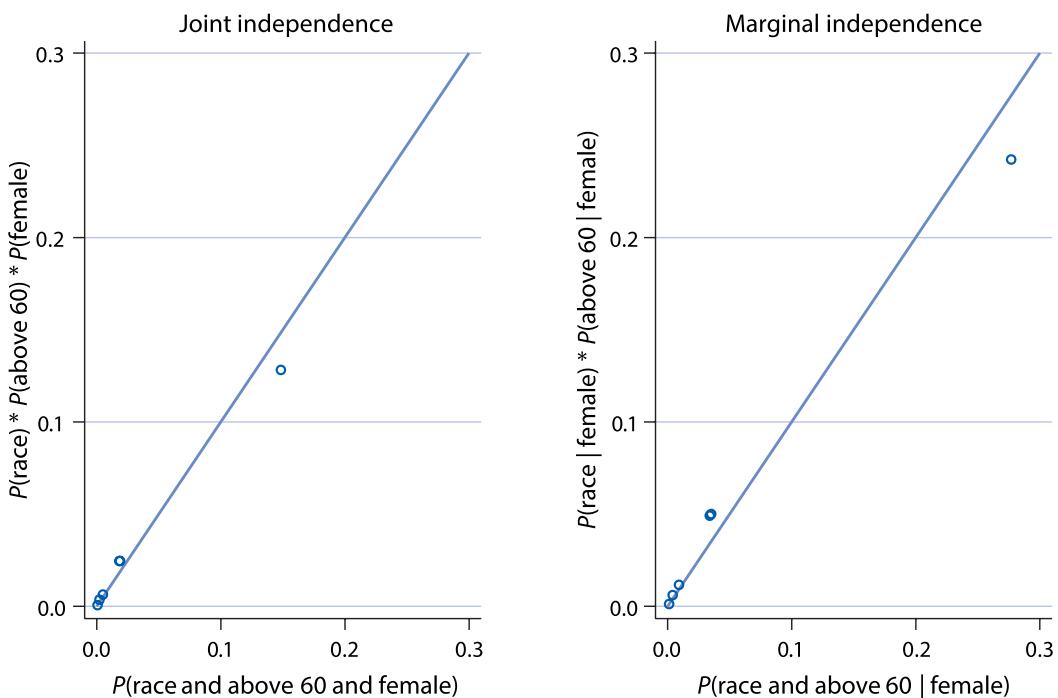
. svmat joint_ind

```

```

.
. * conditional independence, given female
. matrix cond_60rf = joint_60rf / margin_f
.
. matrix c_product_60rf =(joint_rf / margin_f) * (joint_f60 / margin_f)
.
. matrix cond_ind = cond_60rf, c_product_60rf
.
. svmat cond_ind
.
.
. scatter joint_ind2 joint_ind1, msymbol(Oh) || function y=x, range(0 .3) ///
>           xtitle("P(race and above 60 and female)") ///
>           ytitle("P(race) * P(above 60) * P(female)") ///
>           title("Joint independence") legend(off) ///
>           name(joint_ind, replace)
.
. scatter cond_ind2 cond_ind1, msymbol(Oh) || function y=x, range(0 .3) ///
>           xtitle("P(race and above 60 | female)") ///
>           ytitle("P(race | female) * P(above 60 | female)") ///
>           title("Marginal independence") legend(off) ///
>           name(cond_ind, replace)
.
. graph combine joint_ind cond_ind

```



Finally, the well-known *Monty Hall problem* illustrates how tricky conditional probability and independence can be. The problem goes as follows. You are on a game show and must

choose one of three doors, where one conceals a new car and two conceal old goats. After you randomly choose one door, the host of the game show, Monty, opens a different door, which does not conceal a car. Then, Monty asks you if you would like to switch to the (unopened) third door. You will win the new car if it is behind the door of your final choice. Should you switch, or stay with your original choice? Does switching make a difference? Most people think switching makes no difference because after Monty reveals one door with a goat, the two remaining doors have either a goat or a car behind them. Therefore, the chance of winning a car is 50%. However, it turns out that this seemingly sensible reasoning is incorrect.

Let's think about this problem carefully. Consider the strategy of not switching. In this case, your initial choice determines the outcome regardless of what Monty does. Consequently, the probability of winning the car is  $1/3$ . Now, consider the strategy of switching. There are two scenarios. First, suppose that you initially choose a door with the car. The probability of this event is  $1/3$ . Swapping the door in this scenario is a bad choice because you will not win the car. Next, suppose that the door you selected first has a goat. The probability of your initially choosing a door with a goat is  $2/3$ . Then, since Monty opens another door with a goat, the remaining door to which you will switch contains a car. Accordingly, under this scenario, you will always win the car. Thus, switching gives you a probability of winning the car that is twice as high as not switching.

We formalize this logic by applying the rules of probability covered so far. To compute the probability of winning a car given that you switch, we first apply the law of total probability in equation (5.13):

$$\begin{aligned} P(\text{car}) &= P(\text{car} \mid \text{car first})P(\text{car first}) + P(\text{car} \mid \text{goat first})P(\text{goat first}) \\ &= P(\text{goat first}) = \frac{2}{3}. \end{aligned}$$

To see why the second equality holds, notice that if you initially select the door with a car, then switching makes you lose the car, i.e.,  $P(\text{car} \mid \text{car first}) = 0$ . In contrast, if you first pick a door with a goat, then you have a 100% chance of winning a car by switching, i.e.,  $P(\text{car} \mid \text{goat first}) = 1$ .

This rather counter-intuitive problem can also be solved with *Monte Carlo simulations*. For emulating random choice in Stata, we use the `runiformint` function. This function will draw a random integer from a uniform distribution, over an interval we can specify. Because there are three choices (three doors), we specify an interval from 1 to 3, which will yield a value of 1, 2, or 3. The annotated code then traces each of our steps from identifying the selected door using the `: word` function that extracts the  $n$ th word (or element) from our local `doors`, which contains all door options. The `: list` function allows us to remove our selection from the original options. We store the remaining door options in local `remain` and then remove “goat,” which represents Monty’s selection. The no-switch and switch options are stored as string variables that we tabulate after the loops ends.

```
. clear
. set obs 1000
number of observations (_N) was 0, now 1,000
```

```

. * define locals for door options and Monty's choice
. local doors "goat goat car"

. local monty = "goat"

. * create variables to store switch decision
. generate noswitch = ""
(1,000 missing values generated)

. generate switch = ""
(1,000 missing values generated)

.

. forvalues i = 1/ `=_N` {
    2.           * randomly choose the initial door
    .           local first = runiformint(1, 3)
    3.           * identify selection from doors local for each observation
    .           quietly replace noswitch = ``: word `first' of `doors'''' if _n==`i'
    4.           * store selected door in nos local for each observation
    .           quietly local nos = noswitch[`i']
    5.           * remove selected door from doors local, save revised list as remain
    .           quietly local remain : list doors - nos
    6.           * Monty chooses one door with a goat, remove "goat" from remain local
    .           quietly replace switch = ``: list remain - monty'' if _n==`i'
    7. }

```

```
. tabulate noswitch
```

noswitch	Freq.	Percent	Cum.
car	370	37.00	37.00
goat	630	63.00	100.00
Total	1,000	100.00	

```
. tabulate switch
```

switch	Freq.	Percent	Cum.
car	630	63.00	63.00
goat	370	37.00	100.00
Total	1,000	100.00	

### 5.2.3 BAYES' RULE

We discussed different interpretations of probability at the beginning of this chapter. One interpretation, proposed by Reverend Thomas Bayes, was that probability measures one's subjective belief in an event's occurrence. From this Bayesian perspective, it is natural to ask

the question of how we should update our beliefs after observing some data. *Bayes' rule* shows how updating beliefs can be done in a mathematically coherent manner.

**Bayes' rule** is given by

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)} = \frac{P(B | A)P(A)}{P(B | A)P(A) + P(B | A^c)P(A^c)}. \quad (5.21)$$

In this equation,  $P(A)$  is called the **prior probability** and reflects one's initial belief about the likelihood of event  $A$  occurring. After observing the data, represented as event  $B$ , we update our belief and obtain  $P(A | B)$ , which is called the **posterior probability**.

Regardless of whether we interpret probability as subjective belief, Bayes' rule shows mathematically how the knowledge of  $P(A)$  (*prior probability*),  $P(B | A)$ , and  $P(B | A^c)$  yields that of  $P(A | B)$  (*posterior probability*). Bayes' rule is simply the result of rewriting the definition of conditional probability given in equation (5.11) using the law of total probability shown in equation (5.13):

$$P(A | B) = \frac{P(A \text{ and } B)}{P(B)} = \frac{P(B | A)P(A)}{P(B)}.$$

A well-known application of Bayes' rule is the interpretation of medical diagnostic tests, which can have false positives and false negatives (defined in section 4.1.3). Consider the following first-trimester screening test problem. A 35-year-old pregnant woman is told that 1 in 378 women of her age will have a baby with Down syndrome (DS). A first-trimester ultrasound screening procedure indicates that she is in a high-risk category. Of 100 cases of DS, 86 mothers will receive a high-risk result and 14 cases of DS will be missed. Also, there is a 1 in 20 chance for a normal pregnancy to be diagnosed as high risk. Given the result of the screening procedure, what is the probability that her baby has DS? What would the probability be if the result had been negative?

To solve this problem, we first specify the prior probability. Without any testing, the probability that a baby has DS,  $P(DS)$ , is equal to 1/378 or approximately 0.003. The ultrasound screening procedure gives a high-risk result 86% of times when a baby actually has DS. This is called the *true positive rate* of the test and can be expressed as  $P(HR | DS) = 0.86$ , where HR denotes a high-risk result. However, the screening procedure also produces a *false positive rate* of 5%, which can be formally written as  $P(HR | \text{not } DS) = 0.05$ . Using this information, we can apply Bayes' rule to obtain the posterior probability that the baby has DS, given that the woman received a high-risk result, or the *positive predictive value* of the test:

$$\begin{aligned} P(DS | HR) &= \frac{P(HR | DS)P(DS)}{P(HR | DS)P(DS) + P(HR | \text{not } DS)P(\text{not } DS)} \\ &= \frac{0.86 \times \frac{1}{378}}{0.86 \times \frac{1}{378} + 0.05 \times \frac{377}{378}} \approx 0.04. \end{aligned}$$

Similarly, if the woman received a normal pregnancy result, the posterior probability becomes

$$\begin{aligned} P(\text{DS} \mid \text{not HR}) &= \frac{P(\text{not HR} \mid \text{DS})P(\text{DS})}{P(\text{not HR} \mid \text{DS})P(\text{DS}) + P(\text{not HR} \mid \text{not DS})P(\text{not DS})} \\ &= \frac{0.14 \times \frac{1}{378}}{0.14 \times \frac{1}{378} + 0.95 \times \frac{377}{378}} \approx 0.0004. \end{aligned}$$

We see that even when the woman receives a high-risk result, the posterior probability of having a baby with DS is small. This is because DS is a relatively rare disease, as reflected by a small prior probability. As expected, if the woman receives a normal pregnancy result, then the posterior probability becomes even smaller than the prior probability.

We can use Bayes' rule to solve the Monty Hall problem introduced in section 5.2.2. Let  $A$  represent the event that the first door has a car behind it. Define  $B$  and  $C$  similarly for the second and third doors, respectively. Since each door is equally likely to have a car behind it, the prior probabilities are  $P(A) = P(B) = P(C) = 1/3$ . Suppose that we choose the first door and let MC represent the event that Monty opens the third door. We want to know whether switching to the second door increases the chance of winning the car, i.e.,  $P(B \mid \text{MC}) > P(A \mid \text{MC})$ . We apply Bayes' rule after noting that  $P(\text{MC} \mid A) = 1/2$  (Monty chooses between the second and third door with equal probability),  $P(\text{MC} \mid B) = 1$  (Monty has no option but to open the third door, which has a goat), and  $P(\text{MC} \mid C) = 0$  (Monty cannot open the third door, which has a car):

$$\begin{aligned} P(A \mid \text{MC}) &= \frac{P(\text{MC} \mid A)P(A)}{P(\text{MC} \mid A)P(A) + P(\text{MC} \mid B)P(B) + P(\text{MC} \mid C)P(C)} \\ &= \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{2} \times \frac{1}{3} + 1 \times \frac{1}{3} + 0 \times \frac{1}{3}} = \frac{1}{3}, \\ P(B \mid \text{MC}) &= \frac{P(\text{MC} \mid B)P(B)}{P(\text{MC} \mid A)P(A) + P(\text{MC} \mid B)P(B) + P(\text{MC} \mid C)P(C)} \\ &= \frac{1 \times \frac{1}{3}}{\frac{1}{2} \times \frac{1}{3} + 1 \times \frac{1}{3} + 0 \times \frac{1}{3}} = \frac{2}{3}. \end{aligned}$$

Thus, switching doors will give a probability of winning a car that is twice as great as staying with the initial choice.

#### 5.2.4 PREDICTING RACE USING SURNAME AND RESIDENCE LOCATION

This section contains an advanced application of conditional probability and Bayes' rule in the social sciences. Readers may skip this section without affecting their ability to understand the materials in the remainder of the book.

It is often of interest to infer certain unknown attributes of individuals from their known characteristics. We consider the problem of predicting individual race using surname and

residence location.<sup>3</sup> Accurate prediction of individual race is useful, for example, when studying turnout rates among racial groups.

The US Census Bureau releases a list of common surnames with their frequency. For example, the most common surname was “Smith,” with 2,376,206 occurrences, followed by “Johnson” and “Williams,” with 1,857,160 and 1,534,042, respectively. This data set is quite comprehensive, including a total of more than 150,000 surnames that occurred at least 100 times. In addition, the census provides the relative frequencies of individual race within each surname, using a six-category self-reported race measure: non-Hispanic white, non-Hispanic black, non-Hispanic Asian and Pacific Islander, Hispanic origin, non-Hispanic American Indian and Alaskan Native, and non-Hispanic of two or more races. We will combine the last two categories into a single category of non-Hispanic others, so that we have five categories in total. The aggregate information, which can be written as  $P(\text{race} | \text{surname})$ , enables us to predict race given an individual’s surname.

Note that  $P(\text{race})$ ,  $P(\text{race} | \text{surname})$ , and  $P(\text{race and surname})$  are examples of general ways to represent the marginal, conditional, and joint probabilities, respectively. For example,  $P(\text{race})$  represents a collection of marginal probabilities, i.e.,  $P(\text{white})$ ,  $P(\text{black})$ ,  $P(\text{asian})$ ,  $P(\text{hispanic})$ , and  $P(\text{others})$ . Similarly,  $P(\text{race} | \text{surname})$  can be evaluated for any given racial group and surname, such as  $P(\text{black} | \text{Smith})$ . To illustrate the convenience of this general notation, we apply the law of total probability in equation (5.14) to the joint probability of race and surname:

$$P(\text{surname}) = \sum_{\text{race}} P(\text{race and surname}),$$

where the summation is taken over all racial categories (i.e., white, black, asian, hispanic, and others). In terms of the notation used in equation (5.14),  $A$  represents any given surname while  $B_i$  is a racial category. This equality applies to any surname of interest.

This census name list is contained in the data file `names.dta`. Table 5.3 lists the names and descriptions of variables in this census surname list data set.<sup>4</sup>

```
. use names, clear
. describe, short
Contains data from names.dta
obs:      151,671
vars:          7
Sorted by:
29 Feb 2020 16:42
```

The total number of surnames contained in this data set is 151,671. For these surnames, the data set gives the probability of belonging to a particular racial group given a voter’s surname, i.e.,  $P(\text{race} | \text{surname})$ . We begin by using this conditional probability to classify the

<sup>3</sup>This section is in part based on Kosuke Imai and Kabir Khanna (2016). “Improving ecological inference by predicting individual ethnicity from voter registration records.” *Political Analysis*, vol. 24, no. 2 (Spring), pp. 263–272.

<sup>4</sup>To protect anonymity, the Census Bureau does not reveal small race percentages for given surnames. For the sake of simplicity, we impute these missing values by assuming that residual values will be equally allocated to the racial categories with missing values. That is, for each last name, we subtract the sum of the percentages of all races without missing values from 100% and divide the remaining percentage equally among those races that do have missing values.

**Table 5.3.** US Census Bureau Surname List Data.

Variable	Description
surname	surname
count	number of individuals with a specific surname
pctwhite	percentage of non-Hispanic whites among those who have a specific surname
pctblack	percentage of non-Hispanic blacks among those who have a specific surname
pctapi	percentage of non-Hispanic Asians and Pacific Islanders among those who have a specific surname
pcthispanic	percentage of Hispanic origin among those who have a specific surname
pctothers	percentage of the other racial groups among those who have a specific surname

race of individual voters. To validate the accuracy of our prediction of individual race, we use the sample of 10,000 registered voters from Florida analyzed earlier (see Table 5.1). In some Southern states including Florida, voters are asked to self-report their race when registering. This makes the Florida data an ideal validation data set. If the accuracy of a prediction method is empirically validated in Florida, we may use the method to predict individual race in other states where such information is not available.

For matching names between the voter file and census data, we use the `merge` command. The `surname` variable uniquely identifies observations in the `names` data but different voters might have the same surname. Consequently, we specify the `1:m` option when merging the `FLVoters` data into the `names` data.

Going back to the problem of predicting individual racial groups, we remove voters with surnames that do not appear in the census surname list using the `keep()` option. Placing a `3` inside the parentheses tells Stata to only keep observations that appear in both data sets. Alternatively, we could type `match` inside the parentheses. We also drop observations missing values for race.

```
. merge 1:m surname using FLVoters, keep(3)
(note: variable surname was str15, now str20 to accommodate using data's values)

Result # of obs.

not matched 0
matched 8,806 (_merge==3)

.

drop _merge

.

drop if missing(race)
(773 observations deleted)
```

We see that we now focus on 80% of the original sample. We first compute the proportion of voters whose race is correctly classified in each racial category. Race is considered correctly classified if the racial category with the greatest conditional probability  $P(\text{race} | \text{surname})$

is identical to the self-reported race. These represent *true positives* of classification (see Table 4.3).

We calculate the *true positive rate* for each racial group, which represents, for example, the proportion of white voters who are correctly predicted as white. To compute this, we will first recode two of the values of the `race` variable to make them consistent with the corresponding `pct` variables. We then create a variable using `egeen` that holds the maximum value of the `pct` variables.

```
. replace race = "api" if race == "asian"
(140 real changes made)

. replace race = "others" if race == "other" | race == "native"
(281 real changes made)

. egen rmax = rowmax(pctwhite - pctothers)
```

We check whether the highest predicted probability for that voter (`rmax`) is the same as his or her predicted probability of being white (`pctwhite`). If these two numbers are identical, the classification is correct and we change the value of the `correct` variable from 0 to 1. We do the same procedure for the other races, comparing the voter's actual race with the highest predicted probability. To loop over the races, we use the `levelsof` command, which identifies all values of a variable, and we store those values, or levels, in a local macro called `race_cat`. We add the `clean` option to remove quotations. Finally, we tabulate the resulting binary variable by race to obtain the proportion of correct classifications, the true positive rate.

```
. levelsof race, local(race_cat) clean
api black hispanic others white

. generate correct = 0

. foreach v of local race_cat {
    2.         replace correct = 1 if race == ``v'' & pct`v' == rmax
    3. }
(79 real changes made)
(175 real changes made)
(867 real changes made)
(1 real change made)
(5,234 real changes made)

. tabulate correct if race == "white"



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 274   | 4.97    | 4.97   |
| 1       | 5,234 | 95.03   | 100.00 |
| Total   | 5,508 | 100.00  |        |


.
```

The result shows that 95% of white voters are correctly predicted as white. We repeat the same tabulation for black, Hispanic, and Asian voters.

```
. * black
. tabulate correct if race == "black"



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 905   | 83.80   | 83.80  |
| 1       | 175   | 16.20   | 100.00 |
| Total   | 1,080 | 100.00  |        |



. * Hispanic
. tabulate correct if race == "hispanic"



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 157   | 15.33   | 15.33  |
| 1       | 867   | 84.67   | 100.00 |
| Total   | 1,024 | 100.00  |        |



. * Asian
. tabulate correct if race == "api"



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 61    | 43.57   | 43.57  |
| 1       | 79    | 56.43   | 100.00 |
| Total   | 140   | 100.00  |        |


.
```

We find that surname alone can classify 85% of Hispanic voters as Hispanic. In contrast, classification of Asian and black voters is much worse. In particular, only 16% of black voters are correctly classified as African American. The high true positive rate for whites may simply arise from the fact that they far outnumber voters from other racial categories.

We next look at *false positives*. We show the *false discovery rate* for each racial group, which, for example, represents the proportion of voters who are not white among those classified as white. We again use the `correct` variable we created. Now, though, we are interested in the observations where `correct` is equal to 0. We also want to limit the sample to the set of observations where a racial category has the maximum percentage across categories. That is, we restrict the sample to cases where the `rmax` variable equals the racial group percentage variable. We then save our merged data for use later in this chapter.

```

. * whites false discovery rate
. tabulate correct if pctwhite == rmax



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 1,288 | 19.75   | 19.75  |
| 1       | 5,234 | 80.25   | 100.00 |
| Total   | 6,522 | 100.00  |        |



.

. * black false discovery rate
. tabulate correct if pctblack == rmax



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 86    | 32.95   | 32.95  |
| 1       | 175   | 67.05   | 100.00 |
| Total   | 261   | 100.00  |        |



.

. * Hispanic false discovery rate
. tabulate correct if pcthispanic == rmax



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 256   | 22.80   | 22.80  |
| 1       | 867   | 77.20   | 100.00 |
| Total   | 1,123 | 100.00  |        |



.

. * Asian false discovery rate
. tabulate correct if pctapi == rmax



| correct | Freq. | Percent | Cum.   |
|---------|-------|---------|--------|
| 0       | 41    | 34.17   | 34.17  |
| 1       | 79    | 65.83   | 100.00 |
| Total   | 120   | 100.00  |        |



.

. save FL, replace
file FL.dta saved

```

The results show that the false discovery rate is the highest for Asian and black voters, while it is much lower for whites and Hispanics.

**Table 5.4.** Florida Census Data at the Voting District Level.

Variable	Description
county	county census ID of the voting district
VTD	voting district census ID (only unique within the county)
totalpop	total population of the voting district
white	proportion of non-Hispanic whites in the voting district
black	proportion of non-Hispanic blacks in the voting district
api	proportion of non-Hispanic Asian and Pacific Islanders in the voting district
hispanic	proportion of voters of Hispanic origin in the voting district
others	proportion of the other racial groups in the voting district

We attempt to improve the preceding prediction by taking into account where voters live. This approach should be helpful to the extent that there exists residential segregation based on race. In the United States, voter files contain voters' addresses. Using this information, our data set also provides the voting district where each voter lives. In addition, we will utilize the Florida census data, which contains the racial composition of each voting district. The names and descriptions of variables in this census data set, `FLCensusVTD.dta`, are given in table 5.4.

How does the knowledge of residence location improve the prediction of individual race? Whereas the census name data set contains information about the conditional probability  $P(\text{race} | \text{surname})$ , the Florida census data set provides additional information about  $P(\text{race} | \text{residence})$  (proportion of each racial category among residents in a given voting district) and  $P(\text{residence})$  (proportion of residents who live in a given voting district). We wish to combine them and compute the desired conditional probability  $P(\text{race} | \text{surname and residence})$ . Recall that these are general ways to represent marginal, conditional, and joint probabilities. Each expression can be evaluated using a specific racial group, surname, and residential location.

Computing  $P(\text{race} | \text{surname and residence})$  requires Bayes' rule. So far, we have employed Bayes' rule for one event  $A$  conditional on an event  $B$ , but now we need to use Bayes' rule conditional on both  $B$  and another event  $C$ :

$$P(A | B, C) = \frac{P(B | A \text{ and } C)P(A | C)}{P(B | C)},$$

where every probability on the right-hand side is defined conditional on another event  $C$  (see equation (5.21)). Applying this rule yields

$$P(\text{race} | \text{surname and residence}) = \frac{P(\text{surname} | \text{race and residence})P(\text{race} | \text{residence})}{P(\text{surname} | \text{residence})}. \quad (5.22)$$

In this equation, while  $P(\text{race} | \text{residence})$  is available from the Florida census data, the other two conditional probabilities,  $P(\text{surname} | \text{race and residence})$  and  $P(\text{surname} | \text{residence})$ , are not directly given in either the census name data set or the Florida census data set.

To overcome this difficulty, we make an additional assumption that a voter's surname and residence location are independent of each other, given race. This *conditional independence* assumption implies that once we know a voter's race, their residence location does not give us any additional information about their surname. So long as there is no strong geographical concentration of certain surnames in Florida within a racial category, this assumption is reasonable. The assumption is violated, for example, if Hispanic Cubans tend to have distinct names and are concentrated in certain neighborhoods. Unfortunately, our data cannot tell us whether this assumption is appropriate, but we will proceed assuming it is. Applying equation (5.20), the assumption can be written as

$$\begin{aligned} P(\text{surname} \mid \text{race and residence}) &= \frac{P(\text{surname and residence} \mid \text{race})}{P(\text{residence} \mid \text{race})} \\ &= \frac{P(\text{surname} \mid \text{race})P(\text{residence} \mid \text{race})}{P(\text{residence} \mid \text{race})} \\ &= P(\text{surname} \mid \text{race}). \end{aligned} \quad (5.23)$$

The first equality follows from the definition of conditional probability, whereas the second equality is due to the application of equation (5.20).

The assumption transforms equation (5.22) into

$$P(\text{race} \mid \text{surname and residence}) = \frac{P(\text{surname} \mid \text{race})P(\text{race} \mid \text{residence})}{P(\text{surname} \mid \text{residence})}.$$

We should keep this key version of the equation in mind as the one we will ultimately use.

Note that applying the law of total probability defined in equation (5.14) and then invoking the assumption given in equation (5.23), the denominator of equation (5.22) can be written as the following equation, which sums over all racial categories:

$$\begin{aligned} P(\text{surname} \mid \text{residence}) &= \sum_{\text{race}} P(\text{surname} \mid \text{race and residence})P(\text{race} \mid \text{residence}) \\ &= \sum_{\text{race}} P(\text{surname} \mid \text{race})P(\text{race} \mid \text{residence}). \end{aligned} \quad (5.24)$$

In the above equations, we use  $\sum_{\text{race}}$  to indicate summation over all categories of the race variable (i.e., black, white, Asian, Hispanic, and others).

While the census surname list gives  $P(\text{race} \mid \text{surname})$ , the prediction of individual race based on equation (5.22) requires the computation of  $P(\text{surname} \mid \text{race})$ , which is included in both the numerator and the denominator (see equation (5.24)). Fortunately, we can use Bayes' rule to obtain

$$P(\text{surname} \mid \text{race}) = \frac{P(\text{race} \mid \text{surname})P(\text{surname})}{P(\text{race})}. \quad (5.25)$$

The two terms in the numerator of equation (5.25) can be computed using the census name list. We compute  $P(\text{race})$ , which is not included in that data, from the Florida census data by using the law of total probability:

$$P(\text{race}) = \sum_{\text{residence}} P(\text{race} | \text{residence})P(\text{residence}). \quad (5.26)$$

In this equation,  $\sum_{\text{residence}}$  indicates summation over all values of the residence variable (i.e., all voting districts in the data).

To implement this prediction methodology in Stata, we first compute  $P(\text{race})$  using equation (5.26). We do so by calculating a *weighted average* of percentages for each racial category across voting districts with the population of the voting district, which is proportional to  $P(\text{residence})$ , as the weight. To compute the weighted averages, we use the `estpost summarize` command with `[fweight = ]`, where it is necessary to specify a variable of frequency weights. We then save the weighted averages, stored in `e(mean)`, as a transposed matrix to use in the next step. We also save the data for later use.

```
. use FLCensusVTD, clear
. * compute race proportions in Florida
. quietly estpost summarize white black api hispanic others [fweight = totalpop]
. matrix raceprop = e(mean)'
. matrix list raceprop
raceprop[5,1]
      r1
white   .57893423
black   .15164369
api     .0241973
hispanic .22465488
others   .0205699
.
. save FLCensus, replace
file FLCensus.dta saved
```

We can now compute  $P(\text{surname} | \text{race})$  using equation (5.25) and the census name data. We then merge the data back into the FL data set saved earlier, keeping only the observations that matched in both data sets. We also keep only the surname and name variables.

```
. use names, clear
. summarize count
Variable      Obs      Mean      Std. Dev.      Min      Max
count        151,671    1596.357    16338.75      100    2376206
. scalar tot = r(sum)
```

```

.
. * P(surname | race) = P(race | surname) * P(surname) / P(race)
. generate namewhite = (pctwhite / 100) * (count / tot) / matrix(raceprop[1,1])
. generate nameblack = (pctblack / 100) * (count / tot) / matrix(raceprop[2,1])
. generate nameapi = (pctapi / 100) * (count / tot) / matrix(raceprop[3,1])
. generate namehispanic = (pcthispanic / 100) * (count / tot) /
    matrix(raceprop[4,1])
. generate nameothers = (pctothers / 100) * (count / tot) / matrix(raceprop[5,1])
. keep surname name*
. merge 1:m surname using FL
(note: variable surname was str15, now str20 to accommodate using data's values)

      Result          # of obs.

not matched           146,966
      from master       146,966  (_merge==1)
      from using          0  (_merge==2)
matched                8,033  (_merge==3)

.
. drop if missing(race)
(146,966 observations deleted)
.
. drop _merge

```

Next, we compute the denominator of equation (5.22),  $P(\text{surname} | \text{residence})$ , using equation (5.24). To do this, we merge the census data into the voter file data using the county and VTD variables. In the `merge` command, we again set the `keep()` argument to 3 so that nonmatching rows in both data sets will be dropped. Since the census data includes  $P(\text{race} | \text{residence})$  as a variable for each racial category, the merged data set will as well.

```

. merge m:1 county vtd using FLcensus, keep(3)

      Result          # of obs.

not matched           0
matched                8,033  (_merge==3)

.
. drop _merge
.
. generate nameresidence = (namewhite * white) + (nameblack * black) + ///
>     (namehispanic * hispanic) + (nameapi * api) + (nameothers * others)

```

We have now calculated every quantity contained in our key version of equation (5.22):  $P(\text{surname} | \text{race})$ ,  $P(\text{race} | \text{residence})$ , and  $P(\text{surname} | \text{residence})$ . We plug the quantities into

the equation to compute the predicted probability that an individual belongs to a particular race, given his or her surname and residence,  $P(\text{race} | \text{surname}, \text{residence}) = P(\text{surname} | \text{race}) * P(\text{race} | \text{residence}) * P(\text{surname} | \text{residence})$ .

```
. generate prewhite = namewhite * white / nameresidence
. generate preblack = nameblack * black / nameresidence
. generate prehispanic = namehispanic * hispanic / nameresidence
. generate preapi = nameapi * api / nameresidence
. generate preothers = nameothers * others / nameresidence
```

We evaluate the accuracy of this prediction methodology and assess how much improvement knowledge of the voters' location of residence yields. We begin by examining true positives for each race using the same programming commands as before.

```
. egen pre_rmax = rowmax(prewhite - preothers)
. replace race = "api" if race == "asian"
(0 real changes made)

. replace race = "others" if race == "other" | race == "native"
(0 real changes made)

.
. levelsof race, local(race_cat) clean
api black hispanic others white

. generate pre_correct = 0
. foreach v of local race_cat {
    2. replace pre_correct = 1 if race == `v' & pre`v' == pre_rmax
    3. }
(85 real changes made)
(678 real changes made)
(878 real changes made)
(2 real changes made)
(5,187 real changes made)

.
. * white
. tabulate pre_correct if race == "white"

pre_correct | Freq. Percent Cum.
-----|-----
0 | 321 5.83 5.83
1 | 5,187 94.17 100.00
-----|-----
Total | 5,508 100.00
. * blacks
```

```
. tabulate pre_correct if race == "black"
pre_correct | Freq. Percent Cum.
-----|-----
0 | 402 37.22 37.22
1 | 678 62.78 100.00
-----|-----
Total | 1,080 100.00

. * Hispanic
. tabulate pre_correct if race == "hispanic"
pre_correct | Freq. Percent Cum.
-----|-----
0 | 146 14.26 14.26
1 | 878 85.74 100.00
-----|-----
Total | 1,024 100.00

. * Asian
. tabulate pre_correct if race == "api"
pre_correct | Freq. Percent Cum.
-----|-----
0 | 55 39.29 39.29
1 | 85 60.71 100.00
-----|-----
Total | 140 100.00
```

The true positive rate for blacks has jumped from 16% to 63%. Minor improvements are also made for Hispanic and Asian voters. Since African Americans tend to live close to one another in the United States, the location of voters' residences can be informative. For example, according to the census data, among people whose surname is "White," 27% are black. However, once we incorporate the location of their residence, the predicted probability of such individuals being black ranges from 0% to 98%. This implies that we predict some voters to be highly likely black and others highly likely nonblack.

```
. * proportion of blacks among those with surname "White"
. summarize pctblack if surname == "WHITE"
Variable | Obs Mean Std. Dev. Min Max
-----|-----
pctblack | 24 27.38 0 27.38 27.38

. * predicted probability of being black given residence location
. summarize preblack if surname == "WHITE"
Variable | Obs Mean Std. Dev. Min Max
-----|-----
preblack | 24 .2507114 .2938944 .0045884 .9818639
```

Finally, we compute the false positive rate for each race. Again, we are interested in the observations where `pre_correct` is equal to 0.

```
. * white false discovery rate
. tabulate pre_correct if prewhite == pre_rmax
```

pre_correct	Freq.	Percent	Cum.
0	724	12.25	12.25
1	5,187	87.75	100.00
Total	5,911	100.00	

```
. * black false discovery rate
. tabulate pre_correct if preblack == pre_rmax
```

pre_correct	Freq.	Percent	Cum.
0	194	22.25	22.25
1	678	77.75	100.00
Total	872	100.00	

```
. * Hispanic false discovery rate
. tabulate pre_correct if prehispanic == pre_rmax
```

pre_correct	Freq.	Percent	Cum.
0	236	21.18	21.18
1	878	78.82	100.00
Total	1,114	100.00	

```
. * Asian false discovery rate
. tabulate pre_correct if preapi == pre_rmax
```

pre_correct	Freq.	Percent	Cum.
0	42	33.07	33.07
1	85	66.93	100.00
Total	127	100.00	

We find that the false positive rate for whites is significantly reduced. This is in large part due to the fact that many of the black voters who were incorrectly classified as whites using surname alone are now predicted to be black. As an extension, the false positive rate for blacks was also significantly lowered. This example illustrates the powerful use of conditional probability and Bayes' rule.

### 5.3 Random Variables and Probability Distributions

We have so far considered various events including a coin landing on heads, twins being both boys, and a voter being African American. In this section, we introduce the concept of *random variables* and their *probability distributions*, which further widens the scope of mathematical analyses of these events.

#### 5.3.1 RANDOM VARIABLES

A random variable assigns a number to each event. For example, two outcomes of a coin flip can be represented by a binary random variable where 1 indicates landing on heads and 0 denotes landing on tails. Another example is one's income measured in dollars. The values of random variables must represent *mutually exclusive and exhaustive* events. That is, different values cannot represent the same event and all events should be represented by some values. Consider a random variable that represents one's racial group using five unique integers: black = 1, white = 2, hispanic = 3, asian = 4, and others = 5. According to this definition, someone who self-identifies as black and white will be assigned the value of 5 instead of taking the values of 1 and 2 at the same time.

There are two types of random variables, depending on the type of values they take. The first is a *discrete random variable*, which takes a finite (or at most countably infinite) number of distinct values. Examples include categorical or factor variables such as racial groups and number of years of education. The second type is a *continuous random variable*, which takes a value within an interval of the real line. That is, the variable can assume uncountably many values. Examples of continuous random variables include height, weight, and gross domestic product (GDP). The use of random variables, instead of events, facilitates the development of mathematical rules for probability because a random variable takes numeric values. Once we define a random variable, we can formalize a *probability model* using the distribution of the random variable.

A **random variable** assigns a numeric value to each event of the experiment. These values represent mutually exclusive and exhaustive events, together forming the entire sample space. A **discrete random variable** takes a finite or at most countably infinite number of distinct values, whereas a **continuous random variable** assumes an uncountably infinite number of values.

#### 5.3.2 BERNOULLI AND UNIFORM DISTRIBUTIONS

We first consider the simplest example of a *discrete random variable*: a coin flip. For this experiment, we define a *binary random variable*  $X$ , which is equal to 1 if a coin lands on heads and 0 otherwise. In general, a random variable that takes two distinct values is called a *Bernoulli random variable*. Notice that this setup applies to any experiment with two distinct events. Examples include {vote, abstain}, {win election, lose election}, and {correct classification, misclassification}. Thus, whether a voter turns out ( $X = 1$ ) or not ( $X = 0$ ) can be represented by a Bernoulli random variable. Generically, we consider the event  $X = 1$  a success and the event  $X = 0$  a failure. We use  $p$  to denote the probability of success.

The distribution of a discrete random variable can be characterized by the *probability mass function (PMF)*. The PMF  $f(x)$  of a random variable  $X$  is defined as the probability that the random variable takes a particular value  $x$ , i.e.,  $f(x) = P(X = x)$ . That is, given the input  $x$ , which is a specific value of choice, the PMF  $f(x)$  returns as the output the probability that a random variable  $X$  takes that value  $x$ . In the case of a Bernoulli random variable, the PMF takes the value of  $p$  when  $x = 1$  and that of  $1 - p$  when  $x = 0$ . The function is zero at all other values of  $x$ .

Another important function related to probability distribution is the *cumulative distribution function (CDF)*. The CDF  $F(x)$  represents the cumulative probability that a random variable  $X$  takes a value equal to or less than a specific value  $x$ , i.e.,  $F(x) = P(X \leq x)$ . The CDF, therefore, represents the sum of the PMF  $f(x)$  evaluated at all values up to  $x$ . Formally, the relationship between the PMF  $f(x)$  and the CDF  $F(x)$  for a discrete random variable can be written as

$$F(x) = P(X \leq x) = \sum_{k \leq x} f(k),$$

where  $k$  represents all values the random variable  $X$  can take that are less than or equal to  $x$ . That is, the CDF equals the sum of the PMFs. The CDF ranges from 0 to 1 for any random variable, whether continuous or discrete. It is a nondecreasing function because as  $x$  increases, more probability will be added.

The CDF  $F(x)$  for a Bernoulli random variable is simple. It is zero for all negative values of  $x$  because the random variable never assumes any of those values. The CDF then takes the value of  $1 - p$  when  $x = 0$ , which is the probability that  $X$  equals 0. The function stays flat at  $1 - p$  when  $0 \leq x < 1$  because none of these values will be realized. At  $x = 1$ , the CDF equals 1 because the random variable takes either the value of 0 or 1, and stays at this value when  $x \geq 1$  because  $X$  does not take any value greater than 1. Figure 5.5 graphically displays the PMF and CDF of a Bernoulli random variable when  $p = 0.25$ . The open and solid circles represent the exclusion and inclusion of the corresponding points, respectively.

The **probability mass function (PMF)** of a **Bernoulli random variable** with success probability  $p$  is given by,

$$f(x) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \\ 0 & \text{otherwise,} \end{cases}$$

where  $f(1)$  and  $f(0)$  represent the probability of success and failure, respectively. The **cumulative distribution function (CDF)** is given by

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - p & \text{if } 0 \leq x < 1 \\ 1 & \text{if } x \geq 1. \end{cases}$$

We now discuss a *uniform random variable* as a simple example of a *continuous random variable*. A uniform random variable takes every value within a given interval  $[a, b]$  with equal

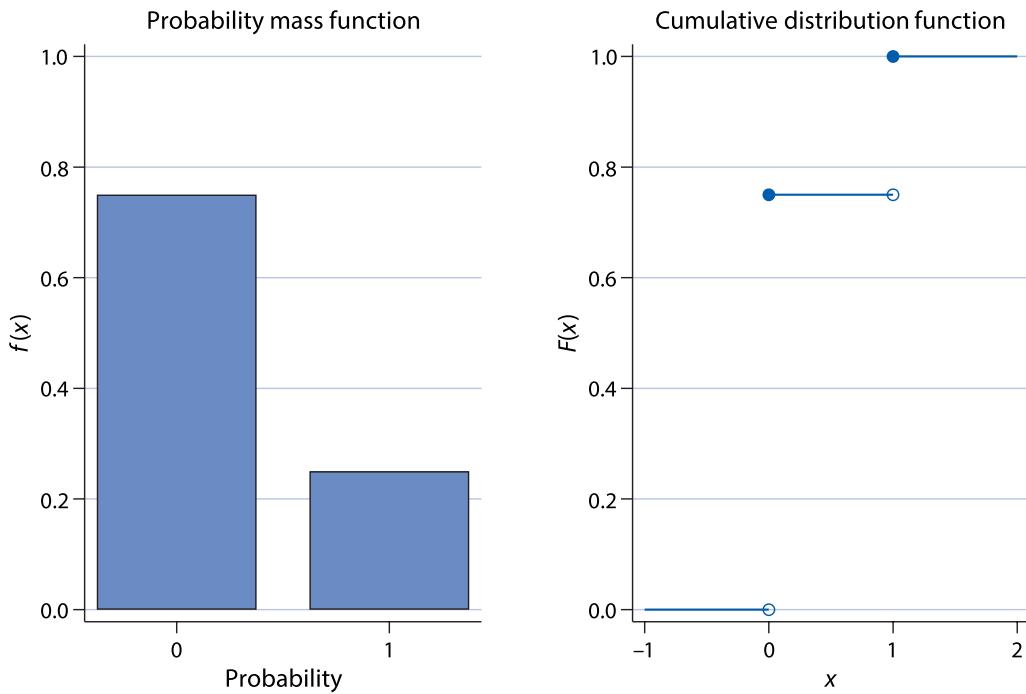


Figure 5.5. The Probability Mass and Cumulative Distribution Functions for a Bernoulli Random Variable. The probability of success is 0.25. The open and solid circles represent the exclusion and inclusion of the corresponding points, respectively.

likelihood. The PMF is not defined for a continuous random variable because this variable assumes an uncountably infinite number of values. Instead, we use the *probability density function* (PDF)  $f(x)$  (or simply, density function), which quantifies the likelihood that a continuous random variable  $X$  will take a specific value  $x$ . We have already seen the concept of *density*, which is used to measure the height of bins in a histogram (see section 3.3.2). The value of the PDF is nonnegative and can be greater than 1. Moreover, like density in histograms, the area under the PDF must sum to 1.

Since each value within the interval is equally likely to be realized, the PDF for the uniform distribution is a flat horizontal line defined by  $1/(b - a)$ . In other words, the PDF does not depend on  $x$  and always equals  $1/(b - a)$  within the interval. The height is determined so that the area below the line equals 1 as required. The left-hand plot of figure 5.6 graphically displays the PDF for a uniform distribution when the interval is set to  $[0, 1]$ .

We can also define the *cumulative distribution function* (CDF) for a continuous random variable. The definition of the CDF is the same as the case of discrete random variables. That is, the CDF  $F(x)$  represents the probability that a random variable  $X$  takes a value less than or equal to a specific value  $x$ , i.e.,  $P(X \leq x)$ . Graphically, the CDF corresponds to the area under the probability density function curve up to the value  $x$  (from negative infinity). Mathematically, this notion can be expressed using integration instead of summation:

$$F(x) = P(X \leq x) = \int_{-\infty}^x f(t) dt.$$

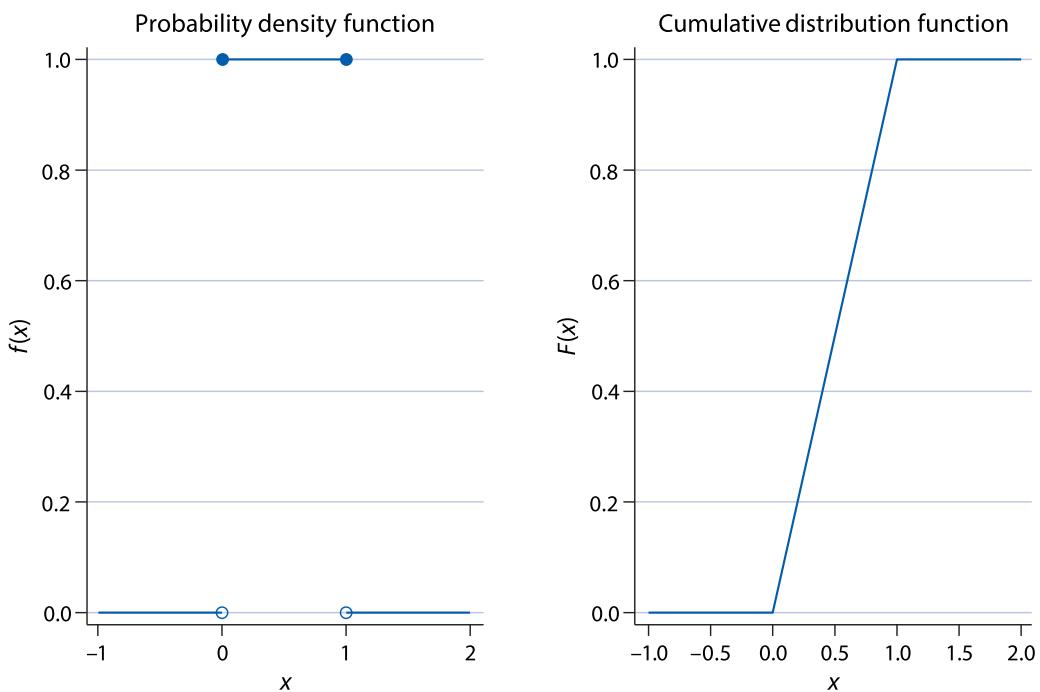


Figure 5.6. The Probability Density and Cumulative Distribution Functions for a Uniform Random Variable. The interval is set to  $[0, 1]$ . The open and solid circles represent the exclusion and inclusion of the corresponding points, respectively.

Since the entire area under the probability density curve has to sum to 1, we have  $F(x) = 1$  when  $x = \infty$ . The CDF for the uniform distribution is shown in the right-hand plot of figure 5.6. In this case, the CDF is a straight line, because the area under the PDF increases at a constant rate. Stata does not offer commands to directly compute the PDF and CDF of a uniform distribution.

The **probability density function** (PDF) of a **uniform random variable** with the interval  $[a, b]$  is given by

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases}$$

The **cumulative probability function** (CDF) is given by

$$F(x) = \begin{cases} 0 & \text{if } x < a, \\ \frac{x-a}{b-a} & \text{if } a \leq x < b, \\ 1 & \text{if } x \geq b. \end{cases}$$

The two distributions we have introduced here share a useful connection. We can use a uniform random variable to generate a Bernoulli random variable. To do this, notice that

under the uniform distribution with unit interval  $[0, 1]$ , the CDF is given by the 45-degree line, i.e.,  $F(x) = x$ . Therefore, the probability that this uniform random variable  $X$  takes a value less than or equal to  $x$  is equal to  $x$  when  $0 \leq x \leq 1$ . In order to generate a Bernoulli random variable  $Y$  with success probability  $p$ , we can first sample a uniform random variable  $X$  and then set  $Y = 1$  when  $X$  is less than  $p$  (similarly, set  $Y = 0$  if  $X \geq p$ ) so that  $Y$  takes a value of 1 with probability  $p$ . To do this *Monte Carlo simulation* in Stata, we use the `runiform()` function to generate a uniform random variable distributed over the interval 0 to 1. We also store the variance of  $y$  as scalar `var_y` to use later.

```
. clear
. set obs 1000
number of observations (_N) was 0, now 1,000
. set seed 12345
. scalar p = .5 // success probabilities
. generate x = runiform()
. list x in 1/6
```

	x
1.	.8296052
2.	.1987135
3.	.1433532
4.	.6322368
5.	.5714322
6.	.4193298

```
. * Bernoulli; turn TRUE / FALSE to 1/0
. generate y = cond(x < p, 1, 0)
. list y in 1/6
```

	y
1.	0
2.	1
3.	1
4.	0
5.	0
6.	1

- . \* close to success probability p, proportion of 1s vs 0s
- . summarize y

Variable	Obs	Mean	Std. Dev.	Min	Max
y	1,000	.508	.5001862	0	1

. scalar var\_y = r(Var)

### 5.3.3 BINOMIAL DISTRIBUTION

The *binomial distribution* is a generalization of the Bernoulli distribution. Instead of a single coin flip, we consider an experiment in which the same coin is flipped independently and multiple times. That is, a binomial random variable can represent the number of times a coin lands on heads in multiple trials of independent coin flips.

More generally, a binomial random variable  $X$  records the number of successes in a total of  $n$  independent and identical trials with success probability  $p$ . In other words, a binomial random variable is the sum of  $n$  *independently and identically distributed* (or *i.i.d.* in short) Bernoulli random variables. Recall that a Bernoulli random variable equals either 1 or 0 with success probability  $p$ . Thus,  $X$  can take an integer value from 0 to  $n$ . Since the binomial distribution is discrete, its PMF can be interpreted as the probability of  $X$  taking a specific value  $x$ . The CDF represents the cumulative probability that a binomial random variable has  $x$  or fewer successes out of  $n$  trials. The PMF and CDF of a binomial random variable are given by the following formulas, which involve combinations (see equation (5.10)). No simple expression exists for the CDF, which is written as the sum of the PMFs.

The probability mass function of a **binomial random variable** with success probability  $p$  and  $n$  trials is given by

$$f(x) = P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}. \quad (5.27)$$

The cumulative distribution function (CDF) can be written as

$$F(x) = P(X \leq x) = \sum_{k=0}^x \binom{n}{k} p^k (1-p)^{n-k},$$

for  $x = 0, 1, \dots, n$ .

Figure 5.7 shows the PMF and CDF when  $p = 0.5$  and  $n = 3$ . For example, we can compute the probability that we obtain two successes out of three trials, which is the height of the third bar in the left-hand plot of the figure:

$$f(2) = P(X = 2) = \binom{3}{2} \times 0.5^2 \times (1 - 0.5)^{3-2} = \frac{3!}{(3-2)!2!} \times 0.5^3 = 0.375.$$

Calculating the PMF of a binomial distribution in Stata is straightforward. The `binomialp()` function takes the number of trials as the first argument, then the number of successes, and the success probability as the final argument.

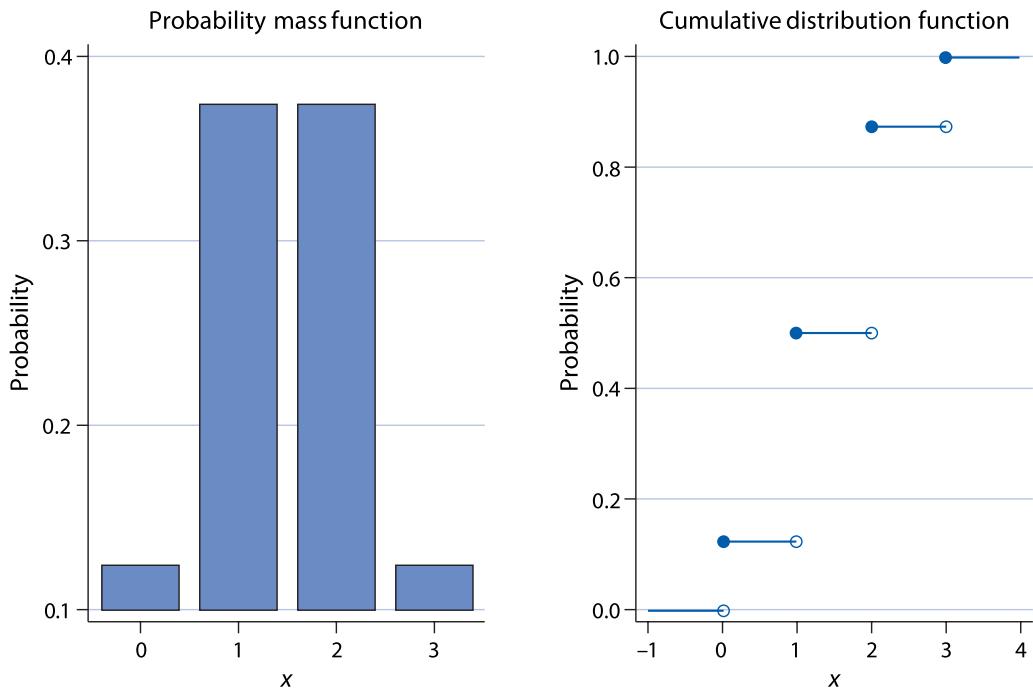


Figure 5.7. The Probability Mass and Cumulative Distribution Functions for a Binomial Random Variable. The success probability is 0.5 and the total number of trials is 3. The open and solid circles represent the exclusion and inclusion of the corresponding points, respectively. *Source:* Adapted from example by Paul Gaborit, <http://texample.net>. (<http://texample.net>.)

```
. * PMF when x = 2, n = 3, p = 0.5
. display binomialp(3, 2, .5)
.375
```

The CDF, shown in the right-hand plot of the figure, is a *step function* where the function is flat and then jumps at each nonnegative integer value. The size of each jump equals the height of the PMF at the corresponding integer value. Using the CDF, we can compute the cumulative probability that we have at most one success out of three trials:

$$F(1) = P(X \leq 1) = P(X = 0) + P(X = 1) = f(0) + f(1) = .125 + .375 = .5.$$

We can compute the CDF of a binomial distribution in Stata using the `binomial()` function.

```
. * CDF when x = 1, n = 3, p = 0.5
. display binomial(3, 1, .5)
.5
```

An intuitive explanation covers why the PMF of a binomial distribution looks like equation (5.27). When we flip a coin  $n$  times, each unique sequence of  $n$  outcomes is equally likely.

For example, if  $n = 5$ , then the event that only the last two coin flips land on tails  $\{HHHTT\}$  is equally as likely as the event that the flips alternate landing on heads and tails  $\{HTHTH\}$ , where we use  $H$  and  $T$  to denote the events that a coin lands on heads and tails, respectively. However, for the binomial distribution only the number of heads matters. As a result, these two events represent the same outcome. We use combinations to count the number of ways we can have  $x$  successes out of  $n$  trials, which is equal to  ${}_n C_x = \binom{n}{x}$ . We multiply this by the probability of  $x$  successes, which is equal to  $p^x$  (because each trial is independent), and the probability of  $n - x$  failures, which is given by  $(1 - p)^{n-x}$  (again because of independence).

As an application of the binomial distribution, consider the probability that one's vote is pivotal in an election. Your vote is pivotal if the election is tied before you cast your ballot. Suppose that in a large population exactly 50% of voters support an incumbent while the other half support a challenger. Further, assume that whether voters turn out or not has nothing to do with their vote choice. Under this scenario, what is the probability that the election ends up with an exact tie? We compute this probability when the number of voters who turn out equals 1,000, then 10,000, and then 100,000. To compute this probability, we can evaluate the PMF of the binomial distribution by setting the success probability to 50% and the size to the total number of voters who turn out. We then evaluate the PMF at exactly half of all voters who turn out. We find that the probability of a tie is quite small, even when the population of voters is evenly divided.

```

. * number of voters who turn out
. foreach n of numlist 1000 10000 100000 {
    2.         display "With `n' voters " binomialp(`n', `=`n' / 2', .5)
    3. }
With 1000 voters .02522502
With 10000 voters .00797865
With 100000 voters .00252313

```

Where does the name “binomial distribution” come from? The name of this distribution is based on the following *binomial theorem*.

The **binomial theorem** shows how to compute the coefficient of each term when expanding the power of a binomial, i.e.,  $(a + b)^n$ . That is, the coefficient for the term  $a^x b^{n-x}$  when expanding  $(a + b)^n$  is equal to  $\binom{n}{x}$ .

For example, according to the binomial theorem, when  $n = 4$ , the coefficient for the term  $a^2 b^2$  when expanding  $(a + b)^4$  is equal to  $\binom{4}{2} = 6$ . This result is confirmed by writing out the entire expansion:

$$(a + b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4. \quad (5.28)$$

These binomial coefficients can be organized as *Pascal’s triangle*, as shown in figure 5.8. The coefficients for the terms resulting from the expansion of  $(a + b)^4$  in equation (5.28) are shown in the fifth row of Pascal’s triangle. More generally, in Pascal’s triangle, the  $x$ th element

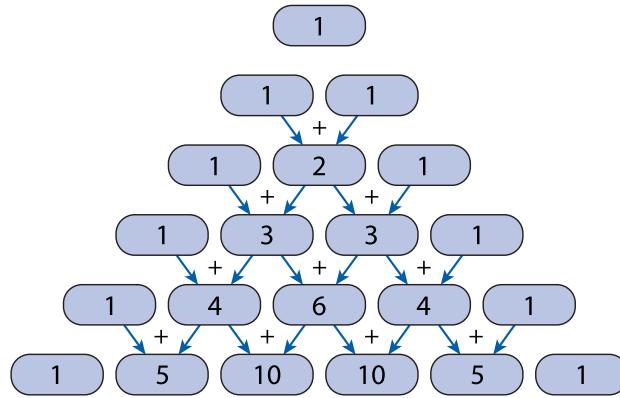


Figure 5.8. Pascal’s Triangle. Binomial coefficients can be represented as Pascal’s triangle where the  $x$ th element of the  $n$ th row returns the binomial coefficient  $\binom{n-1}{x-1}$ . Source: Adapted from example by Paul Gaborit, <http://texexample.net>. (<http://texexample.net>.)

of the  $n$ th row represents the binomial coefficient  $\binom{n-1}{x-1}$ . In addition, as shown in the figure, each element equals the sum of the two elements just above it, leading to a straightforward sequential computation of binomial coefficients. This makes sense because, as an example,  $(a + b)^4$  can be written as the product of  $(a + b)^3$  and  $(a + b)$ ,

$$(a + b)^4 = (a^3 + 3a^2b + 3ab^2 + b^3)(a + b).$$

Here, the coefficient for  $a^2b^2$  is based on the sum of two products, i.e.,  $3a^2b \times b$  and  $3ab^2 \times a$ , and hence is equal to  $6 = 3 + 3$ . In general, to obtain  $x$  success combinations out of  $n$  trials, we consider two scenarios—the last trial ending in a success or ending in a failure—and add the total number of combinations under each scenario:

$$\begin{aligned} \binom{n-1}{x} + \binom{n-1}{x-1} &= \frac{(n-1)!}{x!(n-x-1)!} + \frac{(n-1)!}{(x-1)!(n-x)!} \\ &= (n-1)! \times \frac{(n-x)+x}{x!(n-x)!} = \binom{n}{x}. \end{aligned}$$

The first (second) term corresponds to the scenario where there are  $x$  ( $x - 1$ ) successes out of  $(n - 1)$  trials and the last trial ends in a failure (success).

#### 5.3.4 NORMAL DISTRIBUTION

As another important example of a continuous random variable, we introduce the *normal distribution*. This distribution is also called the *Gaussian distribution*, named after German mathematician Carl Friedrich Gauss. As implied by its name, the normal distribution is special because, as section 5.4.2 will explore, the sum of many random variables from the same distribution tends to follow the normal distribution even when the original distribution is not normal.

A normal random variable can take any number on the real line  $(-\infty, \infty)$ . The normal distribution has two parameters, mean  $\mu$  and standard deviation  $\sigma$ . If  $X$  is a normal random

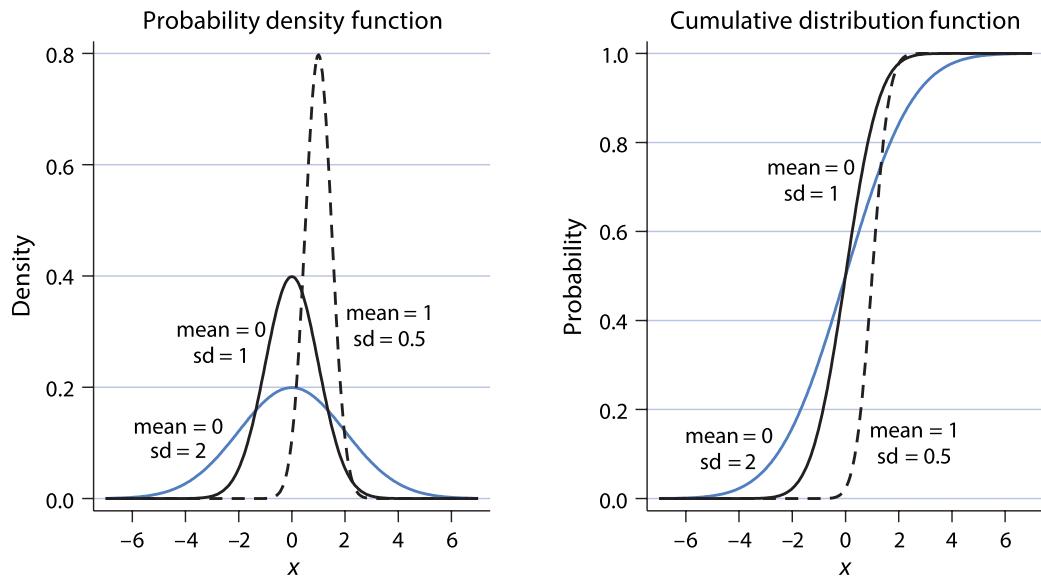


Figure 5.9. The Probability Density and Cumulative Distribution Functions of the Normal Distribution.

variable, we may write  $X \sim \mathcal{N}(\mu, \sigma^2)$ , where  $\sigma^2$  represents the variance (the square of standard deviation). The PDF and the CDF of the normal distribution are given by the following formulas.

The probability density function (PDF) of a **normal random variable** is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

for any  $x$  on the real line. The cumulative probability distribution (CDF) has no analytically tractable form and is given by

$$F(x) = P(X \leq x) = \int_{-\infty}^x f(t) dt = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(t-\mu)^2\right\} dt, \quad (5.29)$$

where  $X \sim \mathcal{N}(\mu, \sigma^2)$  and  $\exp(\cdot)$  is the exponential function (see section 3.4.1). The CDF represents the area under the PDF from negative infinity up to  $x$ .

Figure 5.9 plots the PDF (left-hand plot) and CDF (right-hand plot) for the normal distribution, with three different sets of the mean and standard deviation. The PDF of the normal distribution is bell shaped and centered around its mean, with the standard deviation controlling the spread of the distribution. When the mean is 0 and standard deviation is 1, we have the *standard normal distribution*. The PDF is symmetric around the mean. Different means shift the PDF and CDF without changing their shape. In contrast, a larger standard deviation means more variability, yielding a flatter PDF and a more gradually increasing CDF.

The normal distribution has two important properties. First, adding a constant to (or subtracting it from) a normal random variable yields a normal random variable with an appropriately shifted mean. Second, multiplying (or dividing) a normal random variable by a constant also yields another normal random variable with an appropriately scaled mean and standard deviation. Accordingly, the *z-score* of a normal random variable follows the standard normal distribution. We formally state these properties as follows.

Suppose  $X$  is a normal random variable with mean  $\mu$  and standard deviation  $\sigma$ , i.e.,  $X \sim \mathcal{N}(\mu, \sigma^2)$ . Let  $c$  be an arbitrary constant. Then, the following properties hold:

1. A random variable defined by  $Z = X + c$  also follows a normal distribution with  $Z \sim \mathcal{N}(\mu + c, \sigma^2)$ .
2. A random variable defined by  $Z = cX$  also follows a normal distribution with  $Z \sim \mathcal{N}(c\mu, (c\sigma)^2)$ .

These properties imply that the ***z-score*** of a normal random variable follows the standard normal distribution, which has zero mean and unit variance:

$$\text{z-score} = \frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1).$$

It is important to note that if the data are distributed according to the normal distribution, about two-thirds are within 1 standard deviation from the mean and approximately 95% are within 2 standard deviations from the mean. Let us compute the probability that a normal random variable with mean  $\mu$  and standard deviation  $\sigma$  lies within  $k$  standard deviations from the mean for a positive constant  $k > 0$ . To simplify the computation, consider the *z-score*, which has the standard normal distribution:

$$\begin{aligned} P(\mu - k\sigma \leq X \leq \mu + k\sigma) &= P(-k\sigma \leq X - \mu \leq k\sigma) \\ &= P\left(-k \leq \frac{X - \mu}{\sigma} \leq k\right) \\ &= P(-k \leq Z \leq k), \end{aligned}$$

where  $Z$  is a standard normal random variable. The first equality holds because we subtract  $\mu$  from each term whereas the second equality holds since we divide each term by a positive constant  $\sigma$ .

Thus, the desired probability equals the probability that a standard normal random variable lies between  $-k$  and  $k$ . As illustrated in figure 5.10, this probability can be written as the difference in the CDF evaluated at  $k$  and  $-k$ :

$$P(-k \leq Z \leq k) = P(Z \leq k) - P(Z \leq -k) = F(k) - F(-k),$$

where  $F(k)$  represents the sum of the blue and gray areas in the figure whereas  $F(-k)$  equals the gray area. These results can be confirmed in Stata with the `normal()` function, which



Figure 5.10. The Area under the Probability Density Function Curve of the Normal Distribution. The blue area can be computed as the difference between the cumulative distribution function (CDF) evaluated at  $k$  and  $-k$  (i.e., the gray and blue areas minus the gray area).

evaluates the CDF at its input value. The default is a standard normal distribution with a mean of 0 and standard deviation of 1.

```
.
. * plus minus one standard deviation from the mean
. display normal(1) - normal(-1)
.68268949

. * plus minus two standard deviations from the mean
. display normal(2) - normal(-2)
.95449974
```

The results suggest that, under the standard normal distribution, approximately  $2/3$  are within 1 standard deviation from the mean and about 95% are within 2 standard deviations from the mean. We can also accommodate a mean or standard deviation that does not fall between 0 and 1. We input these values as scalars in Stata and they are transformed into a standard normal random variable within the `normal()` function. Suppose that the original distribution has a mean of 5 and standard deviation of 2, i.e.,  $\mu = 5$  and  $\sigma = 2$ . We can compute the same probabilities as above in the following way.

```
.
scalar mu = 5
scalar sigma = 2
. * plus minus one standard deviation from the mean
. display normal((mu + sigma - mu) / sigma) - normal((mu - sigma - mu) / sigma)
```

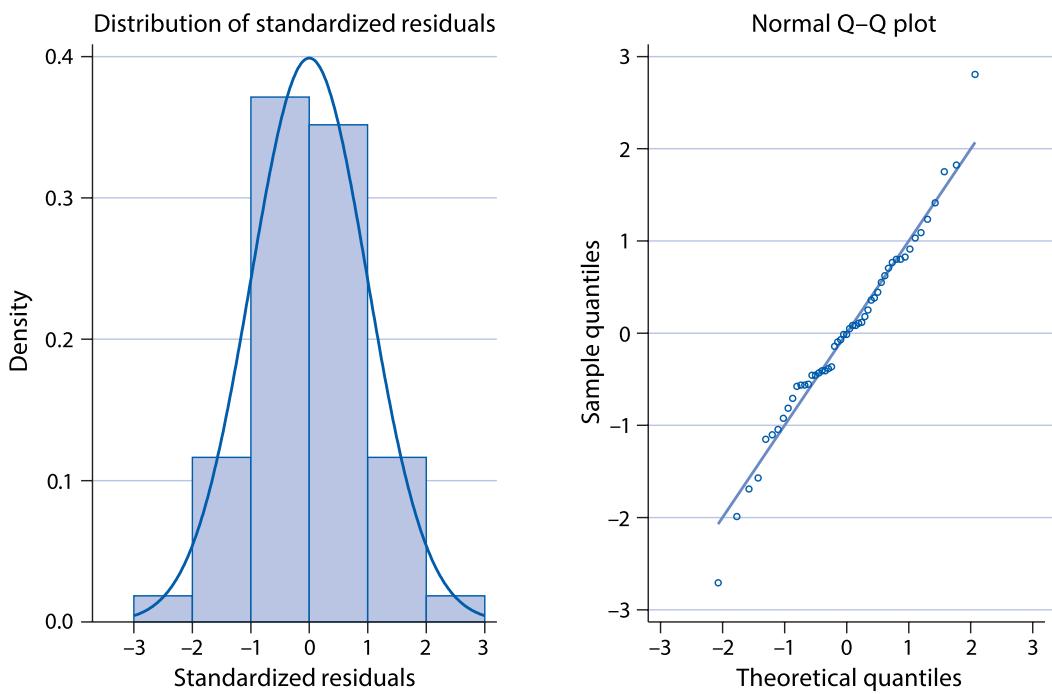
```
.68268949
. * plus minus two standard deviations from the mean
. display normal((mu + 2 * sigma - mu) / sigma) - normal((mu - 2
    * sigma - mu) / sigma)
.95449974
```

As an application of the normal distribution, consider the *regression toward the mean* phenomenon discussed in section 4.2.4. In that section, we presented evidence from US presidential elections demonstrating that in states where Obama received a large share of votes in 2008, he was likely to receive a *smaller* share of votes in 2012. Recall that our regression model used Obama's 2008 statewide vote share to predict his vote share for the same state in the 2012 election. We use the data we saved before and rerun the same regression created on page 157.

```
. use pres0812, clear
. regress obama12z obama08z
```

Source	SS	df	MS	Number of obs	=	51
Model	48.357896	1	48.357896	F(1, 49)	=	1442.99
Residual	1.64210452	49	.033512337	Prob > F	=	0.0000
				R-squared	=	0.9672
				Adj R-squared	=	0.9665
Total	50.0000005	50	1.00000001	Root MSE	=	.18306
obama12z	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
obama08z	.9834419	.0258891	37.99	0.000	.9314157	1.035468
_cons	3.96e-09	.0256341	0.00	1.000	-.0515136	.0515136

We examine the distribution of *residuals* and compare it with the normal distribution (see section 4.2.3 for the definition of residuals). We first present the histogram and overlay the PDF of the normal distribution using the `addplot()` option with function `normalden()` as the new plot. We then use a *quantile-quantile plot* (*Q-Q plot*) to directly compare the distribution of residuals with the normal distribution. The `qnorm` command creates a quantile-quantile plot using the *standard normal distribution*, whose mean is 0 and standard deviation is 1. To make the standard normal distribution and distribution of residuals comparable, we use the `egen` command with the `std()` option to compute the *z-score* of residuals, or *standardized residuals*, whose mean is 0 and standard deviation is 1 (see section 3.7.1). Since residuals always have a mean of 0 (see section 4.2.3), we only need to divide them by their standard deviation to obtain standardized residuals. (Again, we will store the standard deviation as scalar `sd_e` to be used later.)



```

. predict e, resid
. egen e_std = std(e)
. summarize e

      Variable   Obs    Mean    Std. Dev.    Min    Max
e |          51   -6.21e-10   .1812239   -.4906128   .5085258
. scalar sd_e = r(sd)

.
. histogram e_std, addplot(function normalden(x), range(-3 3)) ///
>           xlabel(-3/3) width(1) start(-3) ///
>           xtitle("Standardized residuals") ///
>           title("Distribution of standardized residuals") ///
>           fcolor(none) color(black) legend(off) name(e1, replace)
(bin=6, start=-3, width=1)

. qnorm e_std, xlabel(-3/3) ylabel(-3/3) ///
>           xtitle("Theoretical quantiles") ///
>           ytitle("Sample Quantiles") ///
>           title("Normal Q-Q Plot") msymbol(oh) ///
>           name(e2, replace)

. graph combine e1 e2

```

Both the histogram and Q-Q plot show that the distribution of standardized residuals is remarkably close to the standard normal distribution. Now, consider the following probability model,

Obama's 2012 standardized vote share

$$= 0.983 \times \text{Obama's 2008 standardized vote share} + \epsilon, \quad (5.30)$$

where .983 is the estimated slope coefficient, and the error term  $\epsilon$  follows a normal distribution with mean and standard deviation equal to 0 and .18, respectively. The value of the standard deviation can be found in the results of our already executed `summarize e` command.

This probability model describes a potential data-generating process for Obama's 2012 vote share given his vote share in the previous election. Because both the outcome variable and the predictor are standardized, the intercept is estimated to very close to 0 (and remember that the regression line always goes through the means of the outcome variable and the predictor). We first analyze California where, in 2008, Obama won 61% of the votes or a standardized vote share of .87. According to the proposed probability model, what is the probability that Obama wins a greater share of California votes in 2012? Using the `normal()` function, we can compute the area corresponding to the 2008 vote share under the normal distribution derived for Obama's 2012 votes from the probability model given in equation (5.30). We subtract the results of the `normal()` calculation from 1 in order to compute the probability that Obama wins a *greater* vote share in 2012 than in 2008.

```
. summarize obama08z if state == "CA"
      Variable   Obs    Mean    Std. Dev.    Min    Max
obama08z |       1    .872063          .
. scalar ca08 = r(mean)
. display ca08
.87206304
. scalar ca12 = _b[obama08z] * ca08
. display ca12
.85762331
. display 1 - normal((ca08 - ca12) / sd_e)
.46824629
```

We can see that Obama is somewhat unlikely to win a larger share of California votes in 2012 than he won in 2008. In fact, the probability of this event is only 46.8%. Now, consider Texas where in 2008 Obama received only 44% of the votes or a standardized vote share of  $-0.67$ . Again, under the probability model specified in equation (5.30), we compute the probability that Obama wins a greater share of Texas votes in 2012 than he did in the previous election.

```
. summarize obama08z if state == "TX"

Variable | Obs      Mean   Std. Dev.    Min     Max
-----+-----+-----+-----+-----+-----+
obama08z | 1       -.667812 .       -.667812 -.667812
. scalar tx08 = r(mean)
. display tx08
-.66781205
. scalar tx12 = _b[obama08z] * tx08
. display tx12
-.65675433
. display 1 - normal((tx08 - tx12) / sd_e)
.52432713
```

In the case of Texas, the probability is 52.4%, which is higher than the probability for California. This illustrates the regression toward the mean phenomenon under the probability model based on linear regression with a normally distributed error.

### 5.3.5 EXPECTATION AND VARIANCE

We have introduced several commonly used random variables by defining their PDF/PMF and CDF. These functions completely characterize the distribution of a random variable, but often it is helpful to obtain a more concise summary of a distribution. Previously, we used means and standard deviations in order to measure the center and spread of a distribution. We begin by examining the *expectation*, or mean, of a random variable. We should not confuse this with the *sample mean* discussed earlier in this book. The sample mean refers to the average of a variable in a particular data set, whereas the expectation or *population mean* represents the mean value under a probability distribution. The sample mean fluctuates from one sample to another, but the expectation of a random variable is of a theoretical nature and is fixed given a probability model.

Before we examine the formal definition of expectation, a few examples will prove instructive. Consider a Bernoulli random variable with success probability  $p$  (e.g., a single coin flip with the probability of landing on heads being  $p$ ). What is the expectation? This random variable can take only two values, 0 (tail) and 1 (heads), and so the expectation can be computed as the weighted average of these two values with  $(1 - p)$  and  $p$  (i.e., the PMF) as weights, respectively. Let  $\mathbb{E}(X)$  represent the expectation of a random variable  $X$ . Then, the expectation of a Bernoulli random variable can be computed as

$$\mathbb{E}(X) = 0 \times P(X = 0) + 1 \times P(X = 1) = 0 \times f(0) + 1 \times f(1) = 0 \times (1 - p) + 1 \times p = p. \quad (5.31)$$

Similarly, consider a binomial random variable with success probability  $p$  and size  $n$  (e.g., the number of heads out of  $n$  independent and identical coin flips). This random variable can take any nonnegative integer up to  $n$  (i.e., 0, 1, ...,  $n$ ). The expectation of this binomial random variable is also defined as the weighted average of these values with the weights given

by the corresponding values of the PMF:

$$\mathbb{E}(X) = 0 \times f(0) + 1 \times f(1) + \cdots + n \times f(n) = \sum_{x=0}^n x \times f(x). \quad (5.32)$$

While we use the weighted average to define expectation for a discrete random variable, we need a different way of defining the expectation for a continuous variable. We still compute the weighted average of each value in which the weights are given by the PDF. However, the difference is that a continuous random variable can take an uncountably infinite number of distinct values. This is done through the mathematical operation called *integration*. Readers who are not familiar with calculus can skip the details, but, for example, the expectation of a uniform random variable with interval  $[a, b]$  is calculated as

$$\mathbb{E}(X) = \int_a^b x \times f(x) dx = \int_a^b \frac{x}{b-a} dx = \frac{x^2}{2(b-a)} \Big|_a^b = \frac{a+b}{2}. \quad (5.33)$$

Since each point within the interval is equally likely, the expectation of a uniform random variable equals the midpoint of the interval.

We summarize the general definition of expectation for discrete and continuous random variables.

The **expectation** of a random variable is denoted by  $\mathbb{E}(X)$  and is defined as

$$\mathbb{E}(X) = \begin{cases} \sum_x x \times f(x) & \text{if } X \text{ is discrete,} \\ \int x \times f(x) dx & \text{if } X \text{ is continuous,} \end{cases} \quad (5.34)$$

where  $f(x)$  is the probability mass function or PMF (probability density function or PDF) of the discrete (continuous) random variable  $X$ .

In the definition of expectation, the summation and integration are taken with respect to all possible values of  $X$ . The set of all possible values that  $X$  takes is called the *support* of the distribution. We now introduce the basic rules of the expectation operator  $\mathbb{E}$ .

Let  $X$  and  $Y$  be random variables, and  $a$  and  $b$  be arbitrary constants. The **expectation** is a linear operator that satisfies the following equalities:

1.  $\mathbb{E}(a) = a$
2.  $\mathbb{E}(aX) = a\mathbb{E}(X)$
3.  $\mathbb{E}(aX + b) = a\mathbb{E}(X) + b$
4.  $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$
5. If  $X$  and  $Y$  are independent, then  $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$ . But generally,  $\mathbb{E}(XY) \neq \mathbb{E}(X)\mathbb{E}(Y)$

Applying these rules, we can easily compute the expectation of a binomial random variable. Recall that a binomial random variable  $X$  with success probability  $p$  and size  $n$  is the sum of  $n$  independently and identically distributed (i.i.d.) Bernoulli random variables,  $Y_1, \dots, Y_n$ , with the same success probability  $p$ . This suggests that we can obtain the expectation of the binomial random variable as

$$\mathbb{E}(X) = \mathbb{E}\left(\sum_{i=1}^n Y_i\right) = \sum_{i=1}^n \mathbb{E}(Y_i) = np.$$

This derivation is much more straightforward than the calculation that would be required (i.e., the sum of binomial PMFs evaluated at many values) if we used the definition of expectation given in equation (5.32).

Another useful statistic is the *standard deviation* and its square, *variance*, of a random variable. Both concepts have already been introduced in section 2.6.2. Like the expectation, it is important to distinguish between the standard deviation of a particular sample and the theoretical standard deviation of a random variable. Their interpretations match in that standard deviation is defined as the root mean square (RMS) of deviation from the mean (see section 2.6.2). In the current context, however, we use the expectation, rather than the sample average, to represent the mean.

The **variance** of a random variable  $X$  is defined as

$$\mathbb{V}(X) = \mathbb{E}[\{X - \mathbb{E}(X)\}^2].$$

The square root of  $\mathbb{V}(X)$  is called the **standard deviation**.

Using the basic rules of expectation, we can write the variance as the difference between the expectation of  $X^2$  and the expectation of  $X$ . The expectation of  $X^2$  is called the *second moment*, while the expectation of  $X$ , or the mean, is called the *first moment*:

$$\begin{aligned}\mathbb{V}(X) &= \mathbb{E}[\{X - \mathbb{E}(X)\}^2] \\ &= \mathbb{E}[X^2 - 2X\mathbb{E}(X) + \{\mathbb{E}(X)\}^2] \\ &= \mathbb{E}(X^2) - 2\mathbb{E}(X)\mathbb{E}(X) + \{\mathbb{E}(X)\}^2 \\ &= \mathbb{E}(X^2) - \{\mathbb{E}(X)\}^2.\end{aligned}\tag{5.35}$$

This alternative expression of variance is useful. For example, the variance of a Bernoulli random variable can be derived by noting that  $X = X^2$  regardless of whether  $X$  equals 1 or 0 (because  $1^2 = 1$  and  $0^2 = 0$ ):

$$\mathbb{V}(X) = \mathbb{E}(X) - \{\mathbb{E}(X)\}^2 = p(1 - p).\tag{5.36}$$

This variance is greatest when  $p = 0.5$ . This makes intuitive sense because when  $p$  is smaller, a Bernoulli random variable is more likely to equal 0 and hence has a smaller variance (i.e., less variation).

We can likewise use equation (5.35) to calculate the variance of a uniform random variable with the interval  $[a, b]$ , though readers unfamiliar with integration may ignore the details of the following derivation:

$$\begin{aligned}\mathbb{V}(X) &= \mathbb{E}(X^2) - \{\mathbb{E}(X)\}^2 = \int_a^b \frac{x^2}{b-a} dx - \left(\frac{a+b}{2}\right)^2 \\ &= \frac{x^3}{3(b-a)} \Big|_a^b - \left(\frac{a+b}{2}\right)^2 = \frac{1}{12}(b-a)^2.\end{aligned}\quad (5.37)$$

Like expectation, variance can be approximated through Monte Carlo simulation. Using the set of Bernoulli draws we generated earlier, we compute the sample variance, which should approximate the population variance.

```
. * theoretical variance: p was set to 0.5 earlier
. display p * (1 - p)
.25

. * sample variance using scalar from `y` generated earlier
. display var_y
.25018619
```

Variance has several important properties. For example, since variance involves the expectation of squared distance from the mean, adding a constant to a random variable only shifts the variable and its mean by the same amount without altering its variance. However, the multiplication of a constant and a random variable changes its variance:

$$\mathbb{V}(aX) = \mathbb{E}[\{aX - a\mathbb{E}(X)\}^2] = a^2\mathbb{V}(X). \quad (5.38)$$

We summarize these properties below.

Let  $X$  and  $Y$  be random variables, and  $a$  and  $b$  be arbitrary constants. The **variance** operator  $\mathbb{V}$  has the following properties:

1.  $\mathbb{V}(a) = 0$
2.  $\mathbb{V}(aX) = a^2\mathbb{V}(X)$
3.  $\mathbb{V}(X + b) = \mathbb{V}(X)$
4.  $\mathbb{V}(aX + b) = a^2\mathbb{V}(X)$
5. If  $X$  and  $Y$  are independent,  $\mathbb{V}(X + Y) = \mathbb{V}(X) + \mathbb{V}(Y)$

To compute the variance of a binomial random variable  $X$ , we use its status as the sum of  $n$  independently and identically distributed (i.i.d.) Bernoulli random variables,  $Y_1, Y_2, \dots, Y_n$ , with success probability  $p$ :

$$\mathbb{V}(X) = \mathbb{V}\left(\sum_{i=1}^n Y_i\right) = \sum_{i=1}^n \mathbb{V}(Y_i) = np(1-p).$$

As another example, consider two independent normal random variables  $X$  and  $Y$ . Suppose that  $X$  has mean  $\mu_X$  and variance  $\sigma_X^2$ , whereas  $Y$  has mean  $\mu_Y$  and variance  $\sigma_Y^2$ . We write this setting compactly as  $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$  and  $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ . What is the distribution of  $Z = aX + bY + c$ ? The discussion in section 5.3.4 implies that  $Z$  is also a normal random variable. Using the rules of expectation and variance, we can derive the mean and variance as

$$\begin{aligned}\mathbb{E}(Z) &= a\mathbb{E}(X) + b\mathbb{E}(Y) + c = a\mu_X + b\mu_Y + c, \\ \mathbb{V}(Z) &= \mathbb{V}(aX + bY + c) = a^2\mathbb{V}(X) + b^2\mathbb{V}(Y) = a^2\sigma_X^2 + b^2\sigma_Y^2,\end{aligned}$$

respectively. Therefore, we have  $Z \sim \mathcal{N}(a\mu_X + b\mu_Y + c, a^2\sigma_X^2 + b^2\sigma_Y^2)$ .

### 5.3.6 PREDICTING ELECTION OUTCOMES WITH UNCERTAINTY

We next revisit the prediction of election outcomes using preelection polls. In section 4.1's introduction of the topic, our prediction did not include a measure of uncertainty. However, polling has *sampling variability* because we interview only a fraction of a large population. Suppose that we conduct a preelection poll under the exact same conditions multiple times. Each time, we obtain a representative sample of the target population and yet the sample consists of different voters. This means that the estimated support for a candidate will differ for each sample.

To capture this sampling variability, consider the following probability model. Suppose that the Election Day outcome represents the true proportion of Obama and McCain supporters in the population of voters within each state. We further assume that the fraction of voters who support a third-party candidate is negligible. We therefore focus on the two-party support rate for Obama,  $p_j$ , and McCain,  $1 - p_j$ , within each state  $j$ . The data file `pres08.dta` contains the 2008 US presidential election results (see table 4.1). We first compute the two-party support rate for Obama

```
. use pres08, clear
. * two-party vote share
. generate p = obama / (obama + mccain)
```

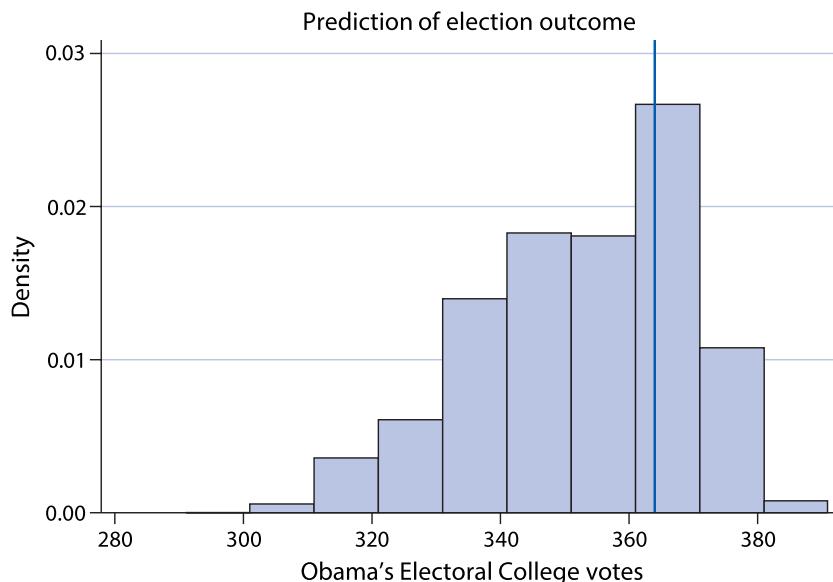
We assume that for each hypothetical sampling, we interview 1,000 voters who are randomly selected from the population. The binomial distribution with success probability  $p$  and size 1,000 within each state is our model for Obama's support estimate based on a preelection poll. Using *Monte Carlo simulation*, we estimate Obama's support within each state, then allocate that state's Electoral College votes to the winning candidate. We will repeat this procedure many times to describe the uncertainty in preelection polling estimates that is due to sampling variability.

To sample from the binomial distribution in Stata, we use the `rbinomial()` function. The first argument is the size of the sample and the second is the success probability. For each success probability, the function will return a new variable of binomial random variable realizations. That is, given  $p_j$  probability of success and  $n = 1,000$  voters, Stata will generate the number of votes for Obama. If a majority of these 1,000 voters support Obama, we assign the state's electoral college votes to Obama. We create a data set of 1,000 observations

and, within our loop, we hold the total electoral votes for Obama from the simulation in a new variable called `obamaev`. Finally, we construct a histogram of these predicted Electoral College votes for Obama. We also add a vertical line to represent the actual number of votes received (364).

```
. set obs 1000
. generate obamaev = .
. forvalues i = 1/1000 {
.     * sample number of votes for Obama in each state
.     quietly generate draws = rbinomial(1000, p)
.     * sum state's Electoral College votes if Obama wins majority
.     quietly summarize ev if draws > 500
.     quietly replace obamaev = r(sum) if _n==`i'
.     * reset draws variable
.     quietly drop draws
. }
```

```
. histogram obamaev, xline(364, lcolor(blue)) ///
>          xtitle("Obama's Electoral College votes") ///
>          title("Prediction of election outcome" width(10) /// 
>          fcolor(none) color(black)
(bin=10, start=291, width=10)
```



We find that all prediction draws are above the winning threshold of 270 votes. While the highest density of the histogram roughly corresponds to the actual number of Electoral

College votes Obama won, the distribution of predictions is skewed. As a result, the mean (352.2) and median (353) values, which we obtain using the `tabstat` command in Stata, are lower than the actual number of Obama's votes.

```
. tabstat obamaev, statistics(mean median)
```

variable	mean	p50
obamaev	352.491	353

We can also analytically compute the expected value of Obama's Electoral College votes under this probability model. Let  $S_j$  represent the number of respondents (among a total of 1,000 respondents) to a preelection poll who express support for Obama in state  $j$ . We use  $v_j$  to denote the number of Electoral College votes for state  $j$ . Then, the expected number of Obama's Electoral College votes is

$$\mathbb{E}(\text{Obama's votes}) = \sum_{j=1}^{51} v_j \times P(\text{Obama wins state } j) = \sum_{j=1}^{51} v_j \times P(S_j > 500). \quad (5.39)$$

To compute this expectation in Stata, we use the `binomial()` function, which evaluates the CDF of the binomial distribution at its input value. As in `binomialp()`, the function takes as its arguments the sample size, number of successes, and the probability of success. Here, we subtract the outcome from 1 because the `binomial()` function evaluates  $P(S_j \leq 500)$ . The threshold 500 is based on the fact that we predict Obama as a winner for a state if more than half of 1,000 respondents support him. Stata's `binomialtail()` function could be used to evaluate  $P(S_j \geq n)$  in which case we would set the number of successes to 501 rather than 500.

```
. * probability of binomial random variable taking greater than n/2 votes
. generate pred_ev = ev * binomialtail(1000, 501, p)
(949 missing values generated)

. tabstat pred_ev, statistics(sum)
```

variable	sum
pred_ev	352.1387

As expected, the analytically derived expected value is close to the value of 352.2 that we approximated using Monte Carlo simulations. Similarly, we can compute the variance of Obama's electoral votes:

$$\begin{aligned}\mathbb{V}(\text{Obama's predicted votes}) &= \sum_{j=1}^{51} \mathbb{V}(v_j \mathbf{1}\{S_j > 500\}) \\ &= \sum_{j=1}^{51} v_j^2 P(S_j > 500) \{1 - P(S_j > 500)\}.\end{aligned}$$

In this derivation,  $\mathbf{1}\{\cdot\}$  represents the *indicator function*, which returns 1 (0) if the statement inside the curly braces is true (false). In addition, the first equality follows from the fact that the variance of the sum of independent random variables equals the sum of their respective variances. We also used the expression for the variance of a Bernoulli random variable given in equation (5.36) because we are evaluating the variance of a Bernoulli random variable  $\mathbf{1}\{S_j > 500\}$ . We compute the variance first with our specified theoretical expression and then with Monte Carlo simulation draws.

```
. * theoretical variance
. generate pb = binomialtail(1000, 501, p)
(949 missing values generated)

. generate variance1 = pb * (1 - pb) * ev^2
(949 missing values generated)

. summarize variance1

      Variable |       Obs        Mean    Std. Dev.      Min      Max
                 |   51    5.270609    17.7855      0    106.1644

. display r(sum)
268.80104

.

. * theoretical standard deviation
. display sqrt(r(sum))
16.395153

. * approximate variance & standard deviation using Monte Carlo draws
. tabstat obamaev, statistics(variance sd)

      variable |     variance        sd
                 |
                 |   obamaev    271.0189   16.46265
```

The result implies that with 1,000 respondents in each state, our poll-based prediction of Obama's Electoral College votes varies from one sample to another. The standard deviation of our prediction is about 16 Electoral College votes. Given that Obama won the election with a much greater margin, this sampling variation did not significantly impact the preelection polls' ability to predict the winner.

## 5.4 Large Sample Theorems

As the final topic of this chapter, we introduce two important probabilistic regularities in large samples. In a wide range of probabilistic models, certain patterns will emerge as the sample size increases. These regularities will quantify the uncertainty of our data analysis in the next chapter. In this section, we discuss two *large sample theorems (asymptotic theorems)*: the law of large numbers and the central limit theorem.

### 5.4.1 THE LAW OF LARGE NUMBERS

The *law of large numbers* states that as the sample size increases, the sample average converges to the expectation or population average.

Suppose that we obtain a random sample of  $n$  independently and identically distributed (i.i.d.) observations,  $X_1, X_2, \dots, X_n$ , from a probability distribution with expectation  $\mathbb{E}(X)$ . The **law of large numbers** states

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i \rightarrow \mathbb{E}(X), \quad (5.40)$$

where we use  $\rightarrow$  as shorthand for convergence.

In the theorem,  $X$  without subscript  $i$  represents a generic random variable, whereas  $X_i$  is the random variable for the  $i$ th observation. Although the precise mathematical meaning of convergence, as well as the precise conditions under which this theorem holds, are beyond the scope of this book, we emphasize that this theorem is applicable to a wide range of probability distributions. Intuitively speaking, the law states that the sample average,  $\bar{X}_n$ , will better approximate the expectation,  $\mathbb{E}(X)$ , as the sample size increases. The law of large numbers is powerful because it can be applied in most settings without knowledge of the underlying probability distribution.

We have already implicitly used the law of large numbers in a variety of contexts. The law of large numbers justifies the use of random sampling in surveys (see section 3.4.1). As we increase the number of randomly sampled respondents, the average response among them becomes closer to the true average of the population. In preelection polls, so long as the sample size is sufficiently large, the sample fraction of those who support Obama approximates the population fraction of voters who are Obama supporters. The law of large numbers enables researchers to talk to a small fraction of randomly sampled individuals in order to infer the opinion of the entire population.

In terms of a probability model, we can think of preelection polling as the sum of independently and identically distributed (i.i.d.) Bernoulli random variables, where a respondent is randomly drawn from a population of Obama supporters and nonsupporters. That is, we define  $X_i$  as an indicator variable of voter  $i$  being an Obama supporter, i.e.,  $X_i = 1$  if voter  $i$  is an Obama supporter and  $X_i = 0$  otherwise. The proportion of Obama supporters in the population is given by  $p$ . Then, the law of large numbers given in equation (5.40) can be directly applied. The sample fraction of Obama's supporters approaches the expectation, or the population proportion of Obama supporters, i.e.,  $\mathbb{E}(X) = p$ .

Additionally, we can rely on the law of large numbers in randomized experiments when computing the difference-in-means between the (randomly divided) treatment and control groups to estimate the average treatment effect (see section 2.4.1). If we consider a population of potential outcomes, as the sizes of the treatment and control groups increase, the sample average of the observed outcome better approximates the expected potential outcome. In other words, we can apply the law of large numbers shown in equation (5.40) by setting  $X$  to each potential outcome,  $Y(1)$  in the treatment group and  $Y(0)$  in the control group.

The law of large numbers can also justify the use of *Monte Carlo simulations*. For example, in the birthday problem described in section 5.1.4, we computed the fraction of simulation trials where at least two birthdays were the same, in order to approximate the true probability of the event occurrence. When applying the law of large numbers shown in equation (5.40), this probability can be written as the expectation by defining a Bernoulli random variable that equals 1 if at least two birthdays match and 0 otherwise. We can then think of the fraction of simulation trials as the sample mean. Adopting the same approach, we solved the Monty Hall problem by computing the fraction of simulation trials in which a contender won a car rather than a goat (see section 5.2.2).

To illustrate the law of large numbers, we conduct a Monte Carlo simulation. We randomly sample from a binomial distribution with success probability  $p = 0.2$  and size  $n = 10$ . We then examine, as the number of binomial draws increases, how the sample mean approaches the expectation, which equals  $E(X) = np = 2$  in this case. To calculate the sample mean after a single draw, two draws, and so on, all the way up to 5,000 draws, we apply the `sum()` function with the `generate` command. This function computes the *cumulative sum*, which combines all values up to and including the current value, for each observation of a variable. Let's say we have three observations for a variable (with values 5, 3, 4), the `sum()` function calculates the cumulative sum (5, 8, 12) for each observation, which we store in another variable (`xsum`). We obtain the desired average for each sample size (5, 4, 4) by dividing the cumulative sum variable by the number of elements used for each summation. We use Stata's `_n` notation to identify the number of elements for each calculation. According to the law of large numbers, a large number of draws should produce a sample mean close to the expectation.

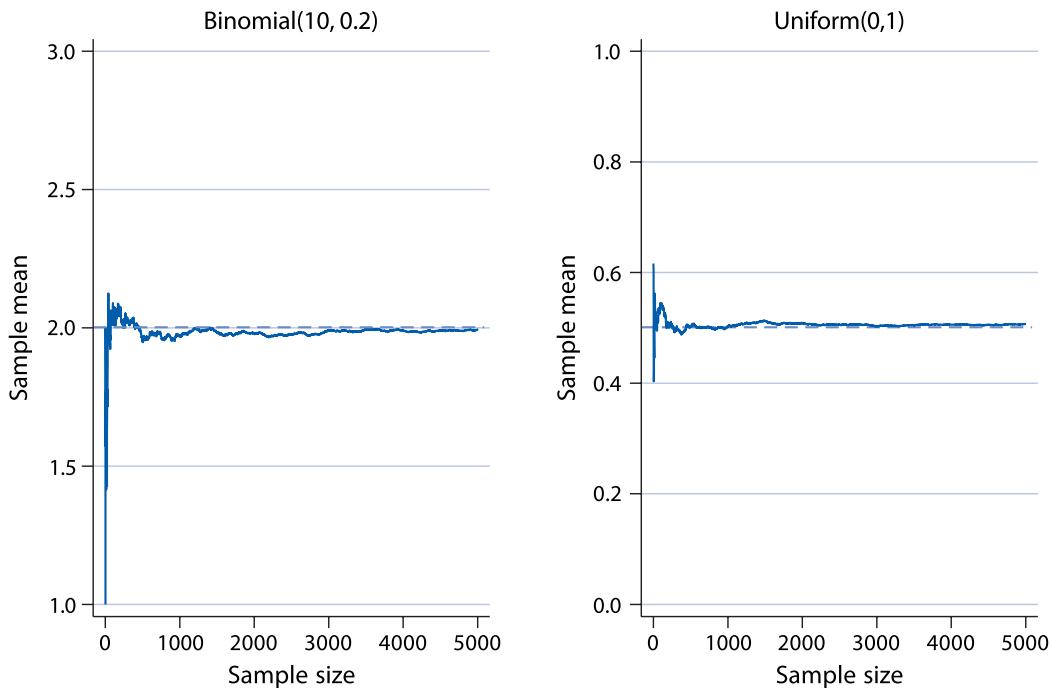
```
. clear
. set obs 5000
number of observations (_N) was 0, now 5,000
. set seed 12345
. generate xbin = rbinomial(10, .2)
. generate xsum = sum(xbin)
. generate meanbin = xsum / _n
```

In addition, we use the uniform distribution as an example of continuous random variables. The `runiform()` function generates a random sample from this distribution.

```
. generate xunif = runiform()
. generate meanunif = sum(xunif) / _n
```

By plotting the results, we see that as the sample size increases, the sample mean approaches the expectation.

```
. generate id = _n
. line meanbin id, yline(2, lpattern(dash)) ///
>      ylabel(1(.5)3) yscale(range(1 3)) xtitle("Sample size") ///
>      ytitle("Sample mean") title("Binomial(10, 0.2)") ///
>      name(bin, replace)
. line meanunif id, yline(.5, lpattern(dash)) ylabel(0(.2)1) ///
>      xtitle("Sample size") ytitle("Sample mean") ///
>      title("Uniform(0,1)") name(unif, replace)
. graph combine bin unif
```



#### 5.4.2 THE CENTRAL LIMIT THEOREM

The law of large numbers is useful but cannot quantify how good the approximation becomes as the sample size increases. In the preceding figure, convergence appears to occur more quickly in the case of the uniform distribution than the binomial distribution. In practice, however, we observe only the sample mean and do not know the expectation. The former is something we compute from the data but the latter is a theoretical concept. Therefore, we need a different tool to know how well our sample mean approximates the expectation.

The *central limit theorem* shows that the distribution of the sample mean approaches the *normal distribution* as the sample size increases. This is a remarkable result because, like the law of large numbers, it applies to a wide range of distributions. The result is useful, as shown in the next chapter, when quantifying the uncertainty of our estimates.

Figure 5.11. The Quincunx as a Machine to Illustrate the Central Limit Theorem.

Before we explain the central limit theorem more formally, we discuss the *quincunx*, invented by Sir Francis Galton who first demonstrated the regression toward the mean phenomenon (section 4.2.4), as a machine that illustrates the theorem. Figure 5.11 presents a picture of a quincunx owned by one of the authors. Red balls are dropped, one at a time, from the tiny hole at the top. The balls, as they fall, bounce off each peg either to its right or left before settling into one of the slots at the bottom of the machine. As seen in the figure, the balls will cluster in the middle, forming a bell-shaped curve that looks like a normal distribution.

Why does the quincunx create a bell-shaped curve? When a ball hits a peg, the ball has a 50–50 chance of bouncing off to its right or left. Although each path from the top to the bottom of the quincunx is equally likely, the ball has more ways to fall into a middle slot than a side slot. More formally, the total number of ways in which a ball reaches a particular slot can be computed using Pascal’s triangle, as shown in figure 5.8. As illustrated in the figure, for example, if there are 5 lines of pegs in the quincunx, there are 20 ways for a ball to fall into the middle two slots.

We can understand the quincunx as a machine that generates a sequence of independently and identically distributed (i.i.d.) binomial random variables  $X$  with success probability 0.5 and size  $n$ , where  $n$  is the number of lines of pegs. Recall that a binomial random variable is the sum of  $n$  i.i.d. Bernoulli random variables. This means that if the central limit theorem holds, then we expect the binomial random variable to approximate the normal random variable as the sample size increases. Here, the sample size refers to the number of lines of pegs, not the number of balls. Increasing the latter reduces the Monte Carlo error. In fact, this is exactly what we observe.

The central limit theorem applies not only to the Bernoulli random variable, but also to other distributions. This is important because in most practical settings we do not know the probability distribution that generates the data. We now more formally state the central limit theorem.

Suppose that we obtain a random sample of  $n$  independently and identically distributed (i.i.d.) observations,  $X_1, X_2, \dots, X_n$ , from a probability distribution with mean  $\mathbb{E}(X)$  and variance  $\mathbb{V}(X)$ . Let us denote the sample average by  $\bar{X}_n = \sum_{i=1}^n X_i/n$ . Then, the **central limit theorem** states

$$\frac{\bar{X}_n - \mathbb{E}(X)}{\sqrt{\mathbb{V}(X)/n}} \xrightarrow{\text{distr.}} \mathcal{N}(0, 1). \quad (5.41)$$

In the theorem,  $\xrightarrow{\text{distr.}}$  indicates “convergence in distribution” as the sample size  $n$  increases.

While formula (5.41) appears complex at first glance, it has a straightforward interpretation. The theorem says that the *z-score* of the sample mean converges in distribution to the standard normal distribution or  $\mathcal{N}(0, 1)$  as the sample size increases. Recall the definition of *z-score* given in equation (3.1). In order to standardize a random variable, we subtract its mean from it and then divide it by its standard deviation. As a result, any *z-score* has zero mean and unit variance.

To show that the left-hand side of formula (5.41) represents the *z-score* of the sample mean, we first note that the expectation of the sample mean  $\bar{X}_n$  is the expectation of the original random variable  $X$ . Using the rules of the expectation operator, we obtain

$$\mathbb{E}(\bar{X}_n) = \mathbb{E}\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}(X_i) = \mathbb{E}(X). \quad (5.42)$$

We next exploit the fact that the variance of two independent random variables equals the sum of their variances. The variance of the sample mean then is given by

$$\mathbb{V}(\bar{X}_n) = \mathbb{V}\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{1}{n^2} \sum_{i=1}^n \mathbb{V}(X_i) = \frac{1}{n} \mathbb{V}(X). \quad (5.43)$$

To derive this expression, we also used formula (5.38). This shows that the denominator of the left-hand side of formula (5.41) represents the standard deviation of the sample mean. Hence, the entire quantity in the left-hand side of formula (5.41) corresponds to the *z-score* of the sample mean.

Monte Carlo simulations can illustrate the central limit theorem. We consider two distributions as examples: the binomial distribution with success probability  $p = 0.2$  and size  $n = 10$ , and the uniform distribution with the range  $[0, 1]$ . Recall that the mean and variance of this binomial distribution are  $np = 10 \times 0.2 = 2$  and  $np(1-p) = 10 \times 0.2 \times (1-0.2) = 1.6$ , respectively. For the uniform distribution, the mean and variance are  $(a+b)/2 = 1/2$  and  $(b-a)^2/12 = 1/12$ , respectively. We use these results to compute the *z-scores* and see whether their distributions can be approximated by the standard normal distribution. To illustrate the quincunx through Monte Carlo simulations, we sample from the Bernoulli distribution or equivalently the binomial distribution with size  $n = 1$ .

To implement these calculations, we can create a program in Stata, using the `program` command. This command allows us to make customized programs that function like other commands. Defining a program follows a simple structure:

```
program name
    command1
    command2
    ...
    commandN
end
```

We assign the program a name then specify the calculations we would like it to make. We then just have to type the name as we would do for other commands, without having to retype the series of calculations contained within the program. To view programs currently in memory, type `program dir`. Programs can be dropped using `program drop`.

We use `program` here to define a program called `zscore`, which produces separate *z-scores* based on the binomial and uniform distributions. In the first line, we specify that our output will be a return result (`rclass`). This means that within our program, Stata will store the results that follow the `return` command. Had our program included estimation commands, such as `regress`, we could specify `eclasse` output and would use the `ereturn` command within our program. We use the `return` command twice—once to store the binomial-based *z-score* (`zbinomial`) and once for the *z-score* based on a uniform distribution (`zunif`). The `syntax` command specifies that `zscore` only accepts integer values as arguments for our sample size input `obs`. Noninteger values will generate an error. We respectively use the `rbinomial` and `runiform` functions to compute *z-scores* based on binomial and uniform distributions.

```
. * define command that calculates z-scores and returns as r-class results
. program zscore, rclass
1.         clear
2.         syntax, obs(integer)
3.         set obs `obs'
4.         * binomial distribution, n = 10 and probability = .2
5.         generate x = rbinomial(10, .2)
6.             summarize x, meanonly
7.             scalar xmean = r(mean)
8.             return scalar zbinomial = (xmean - 2) / sqrt(1.6 / `obs')
9.         * uniform distribution, 0 to 1 interval
10.        generate xu = runiform()
11.            summarize xu, meanonly
12.            scalar xumean = r(mean)
13.            return scalar zunif = (xumean - 0.5) / (sqrt(1 /
14.                (12 * `obs'))))
15. end
```

To run the Monte Carlo simulations, we plug `zscore` into Stata's `simulate` command. This will repeat the calculations defined in our program for as many times as we specify. We run 1,000 replications and store the simulated z-scores found in `r()` as variables `zbin` and `zunif`, which we then plot. The `nodots` option suppresses the display of dots that Stata otherwise shows in the Results console to indicate progress in the number of simulations. This is a useful feature for gauging processing time in simulations that are larger or more complicated than our current example.

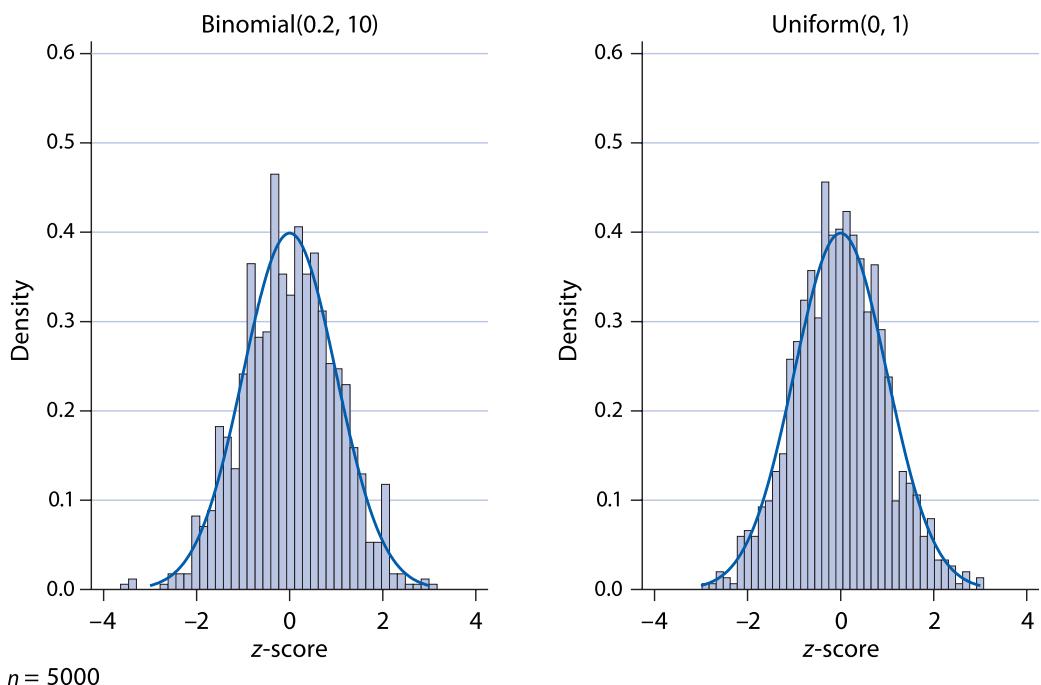
```
. simulate zbin = r(zbinomial) zunif = r(zunif), ///
>           reps(1000) nodots: zscore, obs(5000)

        command: zscore, obs(5000)
        zbin: r(zbinomial)
        zunif: r(zunif)

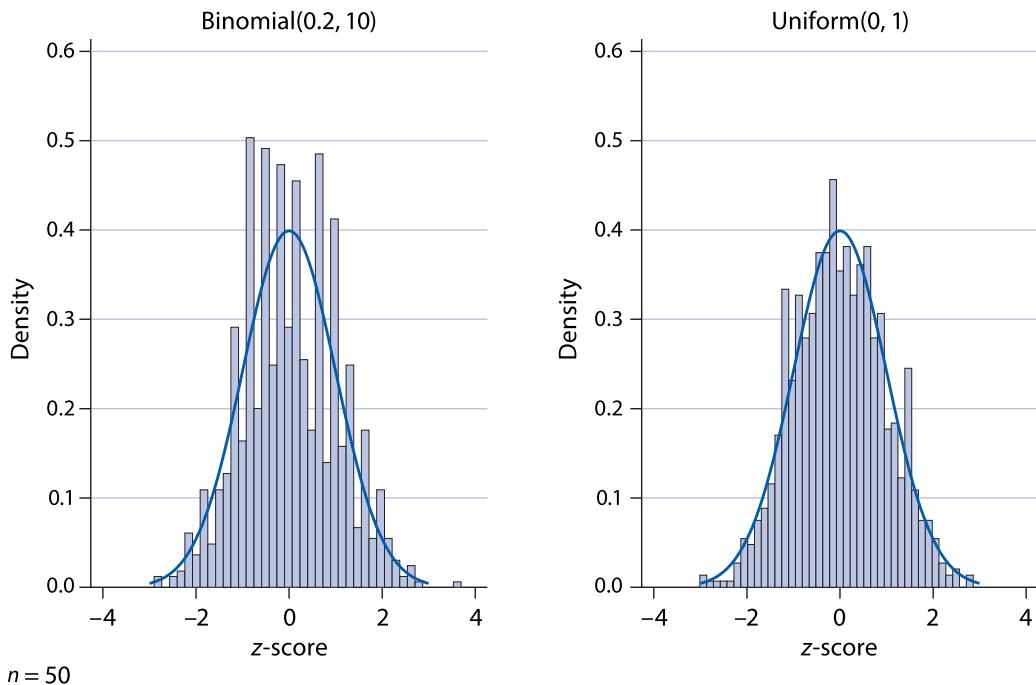
. * plot results
. histogram zbin, bin(40) addplot(function y=normalden(x, 0, 1), ///
>           range(-3 3)) ylabel(0(.1).6) xtitle("z-score") ///
>           title("Binomial(0.2, 10)") legend(off) ///
>           name(bin1, replace)
(bin=40, start=-3.6336105, width=.16994117)

. histogram zunif, bin(40) addplot(function y=normalden(x, 0, 1), ///
>           range(-3 3)) ylabel(0(.1).6) xtitle("z-score") ///
>           title("Uniform(0, 1)") legend(off) ///
>           name(unif1, replace)
(bin=40, start=-2.9760807, width=.15124913)

. graph combine bin1 unif1, note("n = 5000")
```



The above simulations are based on a sample size of 5,000 and we see that the standard normal distribution approximates the distribution of the  $z$ -score well. What about for a smaller sample size? We conduct the same simulation using a sample size of 50. The code is otherwise identical to the one above and therefore omitted.



We observe that the approximation is poorer than before for the binomial distribution, whereas the central limit theorem holds well for the uniform distribution. The theorem does not tell us how large the sample size must be for a good approximation. As shown here, the answer to this question depends on the distribution of the original random variables. Nevertheless, what is incredible about the central limit theorem is that the  $z$ -score of the sample mean converges in distribution to the standard normal distribution *regardless of* the distribution of the original random variable.

## 5.5 Summary

In this chapter, we studied probability. We first introduced two different interpretations of probability, **frequentist** and **Bayesian**. Despite its competing interpretations, probability has a unified mathematical foundation with its basic definition and axioms. We then covered the basic rules of probability, including the **law of total probability**, the definition of **conditional probability**, the concept of **independence**, and **Bayes' rule**. We applied these rules to various problems including the prediction of an individual's race from their surname and residence location.

Next, we examined the concepts of **random variables** and their **probability distributions**. We introduced basic distributions such as **uniform**, **binomial**, and **normal** distributions. These distributions can be characterized by the **probability density function** and **probability mass function** for continuous and discrete random variables, respectively. The **cumulative distribution function** represents the cumulative probability that a random variable takes a

(a) The Enigma machine

(b) Plugboard of the Enigma machine

Figure 5.12. The Enigma Machine and Its Plugboard. *Source:* Wikipedia

value less than or equal to a specified value. Using the probability mass and density functions, we showed how to compute the **expectation** and **variance** of a random variable. We used these tools to quantify the sampling uncertainty regarding the polling prediction of election results.

Lastly, we discussed the two fundamental large sample approximation theorems. The power of these theorems is that they can be applied to the sample mean of virtually any random variable given a sufficient sample size. The **law of large numbers** states that the sample mean approaches the expectation or the population mean as the sample size increases. This justifies the use of the sample mean as an estimator of the population mean in survey sampling and randomized experiments. The **central limit theorem** states that the z-score of the sample mean is approximately distributed according to the standard normal distribution. In the next chapter, we will use these large sample theorems to quantify the degree of uncertainty regarding the empirical conclusions drawn from our data analyses.

## 5.6 Exercises

### 5.6.1 THE MATHEMATICS OF ENIGMA

The Enigma machine is the most famous cipher machine to date. Nazi Germany used it during World War II to encrypt messages so that enemies could not understand them. The story of the British cryptanalysts who successfully deciphered Enigma has become the subject of multiple movies (*Enigma*, 2001; *The Imitation Game*, 2014). In this exercise, we will focus our attention on a simplified version of the Enigma machine, which we name “Little Enigma.” Like the real Enigma machine shown in Figure 5.12a, this machine consists of two key components. First, the Little Enigma machine has five different *rotors*, each of which comes with 10 pins with numbers ranging from 0 to 9. Second, as shown in figure 5.12b, the

*plugboard* contains 26 holes, corresponding to the 26 letters of the alphabet. In addition, 13 cables connect all possible pairs of letters. Since a cable has two ends, one can connect, for example, the letter A with any of the other 25 letters present in the plugboard.

To either encode a message or decode an encrypted message, one must provide the Little Enigma machine with a correct five-digit passcode to align the rotors and a correct configuration of the plugboard. The rotors are set up just like many combination locks. For example, the passcode 9–4–2–4–9 means that five rotors display the numbers 9, 4, 2, 4, and 9 in that order. Further, the 13 cables connecting the letters in the plugboard must be appropriately configured. The purpose of the plugboard is thus to scramble the letters. In a case where B is connected to W, the Little Enigma machine will switch B with W and W with B to encode a message or decode an encoded message. Summarizing the process, a sender types a message on the keyboard, the plugboard scrambles the letters, and the message is sent in its encrypted form. A receiver decodes the encrypted message by retyping it on a paired Little Enigma machine that has the same passcode and plugboard configuration.

1. How many different five-digit passcodes can be set out of the five rotors?
2. How many possible configurations does the plugboard provide? In other words, how many ways can 26 letters be divided into 13 pairs?
3. Based on the previous two questions, what is the total number of possible settings for the Little Enigma machine?
4. Five cryptanalytic machines have been developed to decode 1,500 messages encrypted by the Little Enigma machine. The table below presents information on the number of messages assigned to each machine and the machine's failure rate (i.e., the percentage of messages the machine was unable to decode). Aside from this information, we do not know anything about the assignment of each message to a machine or whether the machine was able to correctly decode the message.

Machine	Number of messages	Failure rate (%)
Banburismus	300	10
Bombe	400	5
Herivel tip	250	15
Crib	340	17
Hut 6	210	20

Suppose that we select one message at random from the pool of all 1,500 messages but found out this message was not properly decoded. Which machine is most likely responsible for this mistake?

### 5.6.2 A PROBABILITY MODEL FOR BETTING MARKET ELECTION PREDICTION

Earlier in this chapter, we used preelection polls with a probability model to predict Obama's electoral vote share in the 2008 US election. In this exercise, we will apply a similar procedure to the Intrade betting market data analyzed in exercise 4.5.1.<sup>5</sup> The 2008 Intrade data are available as `intrade08.dta`. The variable names and descriptions of this data set are available in table 4.9. Recall that each row of the data set represents daily trading information about the contracts for either the Democratic or Republican Party nominee's victory in a particular state. The 2008 election results data are available as `pres08.dta`, with variable names and descriptions appearing in table 4.1.

1. We analyze the contract of the Democratic Party nominee winning a given state  $j$ . Recall from section 4.5.1 that the data set contains the contract price of the market for each state on each day  $i$  leading up to the election. We will interpret `priced` as the probability  $p_{ij}$  that the Democrat would win state  $j$  if the election were held on day  $i$ . To treat `priced` as a probability, divide it by 100 so it ranges from 0 to 1. How accurate is this probability? Using only the data from the day before Election Day (November 4, 2008) within each state, compute the expected number of electoral votes Obama is predicted to win and compare it with the actual number of electoral votes Obama won. Briefly interpret the result. Recall that the actual total number of electoral votes for Obama is 365, not 364, which is the sum of electoral votes for Obama based on the results data. The total of 365 includes a single electoral vote that Obama garnered from Nebraska's 2nd Congressional District. McCain won Nebraska's four other electoral votes because he won the state overall.
2. Next, using the same set of probabilities used in the previous question, simulate the total number of electoral votes Obama is predicted to win. Assume that the election in each state is a Bernoulli trial where the probability of success (Obama winning) is  $p_{ij}$ . Display the results using a histogram. Add the actual number of electoral votes Obama won as a solid line. Briefly interpret the result.
3. In prediction markets, people tend to exaggerate the likelihood that the trailing or "long shot" candidate will win. This means that candidates with a low (high)  $p_{ij}$  have a true probability that is lower (higher) than their predicted  $p_{ij}$ . Such a discrepancy could introduce bias into our predictions, so we want to adjust our probabilities to account for it. We do so by reducing the probability for candidates who have a less than 0.5 chance of winning, and increasing the probability for those with a greater than 0.5 chance. We will calculate a new probability  $p_{ij}^*$  using the following formula proposed by a researcher:  $p_{ij}^* = \Phi(1.64 \times \Phi^{-1}(p_{ij}))$  where  $\Phi(\cdot)$  is the CDF of the standard normal random variable and  $\Phi^{-1}(\cdot)$  is its inverse, the quantile function. The Stata functions `normal()` and `invnormal()` can be used to compute  $\Phi(\cdot)$  and  $\Phi^{-1}(\cdot)$ , respectively. Plot  $p_{ij}$ , used in the previous questions, against  $p_{ij}^*$ . In addition, plot this function itself as a line. Explain the nature of the transformation.

<sup>5</sup>This exercise is based on David Rothschild (2009). "Forecasting elections: Comparing prediction markets, polls, and their biases." *Public Opinion Quarterly*, vol. 73, no. 5, pp. 895–916.

4. Using the new probabilities  $p_{ij}^*$ , repeat Questions 1 and 2. Do the new probabilities improve predictive performance?
5. Compute the expected number of Obama's electoral votes using the new probabilities  $p_{ij}^*$  for each of the last 120 days of the campaign. Display the result as a time series plot. Briefly interpret the plot.
6. For each of the last 120 days of the campaign, conduct a simulation as done in Question 2 using the new probabilities  $p_{ij}^*$ . Compute the quantiles of Obama's electoral votes at 2.5% and 97.5% for each day. Represent the range from 2.5% to 97.5% for each day as a vertical line, using a loop. Also, add the estimated total number of Obama's electoral votes across simulations. Briefly interpret the result. Hint: You can use `ereturn post` to return a row vector in a program. The command `xpose` will transpose numeric variables into observations.

## Chapter 6

---

# Uncertainty

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

—Albert Einstein, *Geometry and Experience*

Thus far, we have studied various data analysis techniques that can extract useful information from data. We have used these methods to draw causal inferences, measure quantities of interest, make predictions, and discover patterns in data. One important remaining question, however, is how certain we can be of our empirical findings. For example, if in a randomized controlled trial the average outcome differs between the treatment and control groups, when is this difference large enough for us to conclude that the treatment of interest affects the outcome, on average? Did the observed difference result from chance? In this chapter, we consider how to separate signals from noise in data by quantifying the degree of uncertainty. We do so by applying the laws of probability introduced in the previous chapter. We cover several concepts and methodologies to formally quantify the level of uncertainty. These include bias, standard errors, confidence intervals, and hypothesis testing. Finally, we describe ways to make inferences from linear regression models with measures of uncertainty.

### 6.1 Estimation

In earlier chapters, we showed how to infer public opinion in a population through survey sampling (chapter 3) and estimate causal effects through randomized controlled trials (chapter 2). In these examples, researchers want to estimate the unknown value of a quantity of interest using observed data. We refer to the quantity of interest as a *parameter* and the method to compute its estimate as an *estimator*. For example, in the analysis of survey data presented in chapter 4, we are interested in estimating the proportion of Obama supporters in the population of American voters (parameter) based on a relatively small number of survey respondents (data). We use the sample proportion of Obama supporters as our estimator. Similarly, in randomized controlled trials, the average outcome difference between the treatment and control groups represents an estimator for the average causal effect, which is our parameter.

How good is our estimate of the parameter? This is a difficult question to answer because we do not know the true value of the parameter. However, it turns out that we can characterize how well the estimator will perform over hypothetically repeated sampling. This section shows how statistical theory can help us investigate the performance of the estimators we used in the earlier parts of the book.

### 6.1.1 UNBIASEDNESS AND CONSISTENCY

Consider a survey for which a certain number of respondents are selected from a population using the *simple random sampling* procedure. Simple random sampling implies that each individual in the population is equally likely to be selected into a sample. As discussed in chapter 3, such random sampling benefits us by producing a representative sample of a target population (see section 3.4.1).

To give further context to these ideas, recall the preelection polling example in the 2008 US presidential election (see section 4.1.3). In that example, our parameter was the proportion of voters in the population of American voters that supported Obama. We used simple random sampling to obtain a representative sample of  $n$  voters from the population. The survey asked whether each of the respondents supported Obama or not. We used the sample proportion of those who supported Obama as our estimate of the population proportion of Obama supporters.

To formalize the content of the previous paragraph, let  $p$  denote the population proportion of Obama supporters. We use a random variable  $X$  to represent a response to the question. If voter  $i$  supports (does not support) Obama, then we denote this observation with  $X_i = 1$  ( $X_i = 0$ ). Since each respondent is sampled independently from the same population, we can assume that  $\{X_1, X_2, \dots, X_n\}$  are independently and identically distributed (i.i.d.) Bernoulli random variables with success probability  $p$  (see section 5.3.2). Our estimator is the sample proportion,  $\bar{X}_n = \sum_{i=1}^n X_i/n$ , which we use to estimate the unknown parameter  $p$ . The specific value of this estimator we obtain from our sample represents the estimate of  $p$ .

How good is this estimate? Ideally, we would like to compute the *estimation error*, which is defined as the difference between our estimate and the truth:

$$\text{estimation error} = \text{estimate} - \text{truth} = \bar{X}_n - p.$$

However, the estimation error can never be computed because we do not know  $p$ . In fact, if we know the truth, there is no need to estimate the parameter in the first place!

While we never know the size of the estimation error specific to our sample, it is sometimes possible to compute the *average* magnitude of the estimation error. To do this, we consider the hypothetical scenario of conducting the same preelection poll infinitely many times in exactly the same manner. This scenario is purely hypothetical because in reality we obtain only one sample and can never conduct sampling in an identical manner multiple times. Under this scenario, each hypothetical poll would draw a different set of  $n$  voters from the sample population and yield a different proportion of sampled voters who express support for Obama. This means that the sample proportion, represented by a random variable  $\bar{X}_n$ , would take a different value for each poll. As a result, the estimation error would also differ from one poll to another and hence is a random variable.

More formally, the sample proportion can be considered as a random variable that has its own distribution over the repeated use of simple random sampling. This distribution is

called the *sampling distribution* of the estimator. In this particular example, each hypothetical sample is drawn independently from the same population. Therefore, the sample proportion,  $\bar{X}_n$ , is a binomial random variable, divided by  $n$ , with success probability  $p$  and size  $n$  where  $n$  represents the number of respondents in a poll (recall from section 5.3.3 that the sum of i.i.d. Bernoulli random variables is a binomial random variable).

We now compute the average estimation error or *bias* over this repeated simple random sampling procedure using the concept of *expectation* (see section 5.3.5). Under the binomial model, the success probability equals  $p$ . We can then show that the bias, or the average estimation error, of the sample mean is zero:

$$\text{bias} = \mathbb{E}(\text{estimation error}) = \mathbb{E}(\text{estimate} - \text{truth}) = \mathbb{E}(\bar{X}_n) - p = p - p = 0.$$

This result implies that the sample proportion under simple random sampling is an *unbiased* estimator for the population proportion. That is, while the sample proportion based on a specific sample may deviate from the population proportion, it gives, on average, the right answer. More precisely, if we were to conduct the same preelection poll infinitely many times under identical conditions, the average of the sample proportions of Obama supporters would exactly equal their population proportion. Thus, unbiasedness refers to the accuracy of the average estimate over repeated sampling rather than the accuracy of an estimate based on the observed data.

Similar logic applies to nonbinary variables. We can show that the expectation of the sample mean equals the population average so long as each survey respondent is randomly sampled from a large population. An example of a nonindependent sampling procedure is respondent-driven sampling, in which one respondent introduces another respondent to the interviewer. Using the fact that expectation is a linear operator (see section 5.3.5), we obtain the following general result for the sample mean:

$$\mathbb{E}(\bar{X}_n) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}(X_i) = \mathbb{E}(X). \quad (6.1)$$

The final equality follows because each of the  $n$  observations is randomly sampled from the same population whose mean is denoted by  $\mathbb{E}(X)$ . Hence, regardless of the distribution of a variable, random sampling provides a way to use the sample average as an unbiased estimator of the population mean. Equation (6.1) shows that random sampling eliminates bias.

In general, random sampling plays an essential role in obtaining an unbiased estimate. In the absence of random sampling or other ways to obtain a representative sample, it is difficult to estimate a population characteristic without bias. For example, item and unit *nonresponse*, discussed in section 3.4.2, can yield biased estimates.

In section 5.4.1, we introduced the *law of large numbers*, which states that as sample size increases, the sample mean converges to the population mean. In the current context, this implies that the estimation error, which is the difference between the sample mean and the population mean, becomes smaller as the sample size increases. The estimator is said to be *consistent* if it converges to the parameter as the sample size goes to infinity. The discussion so far implies that the sample mean is a good estimator for the population mean because it is an unbiased and consistent estimator of the population mean. That is, the sample mean on average correctly estimates the population mean, and the estimation error decreases as the sample size increases.

An estimator is said to be **unbiased** if its expectation equals the parameter. An estimator is said to be **consistent** if it converges to the parameter as the sample size increases. For example, the sample average  $\bar{X}_n = \sum_{i=1}^n X_i/n$  is unbiased and consistent for the population mean  $\mathbb{E}(X)$  under simple random sampling:

$$\mathbb{E}(\bar{X}_n) = \mathbb{E}(X) \quad \text{and} \quad \bar{X}_n \rightarrow \mathbb{E}(X).$$

We next show that the difference-in-means estimator used to analyze *randomized controlled trials* (see section 2.4) is unbiased for the average treatment effect. Suppose that we have a sample of  $n$  units for which we conduct a randomized experiment. This experiment features a single binary treatment  $T_i$ , which equals 1 if unit  $i$  receives the treatment and 0 if the unit is assigned to the control group. We randomly choose  $n_1$  units out of this sample and assign them to the treatment group, and the remaining  $n - n_1$  units belong to the control group. This treatment assignment procedure is called *complete randomization*, which fixes a priori the total number of units that receive the treatment. In contrast, *simple randomization* randomly assigns treatment to each unit independently, and so the total number of treated units will vary from one randomization to another. Under complete randomization, there exists a total of  $\binom{n}{n_1}$  ways of assigning  $n_1$  units to the treatment group and the remaining units to the control group (see section 5.1.5 for the definition of combinations). Each of these treatment assignment combinations is equally likely but only one of them is realized.

The first parameter we consider, the *sample average treatment effect* (SATE), is defined in equation (2.1) and reproduced here:

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

In this equation,  $Y_i(1)$  and  $Y_i(0)$  are the potential outcomes under the treatment and control conditions for unit  $i$ , respectively. As discussed in section 2.3,  $Y_i(1)$  ( $Y_i(0)$ ) represents the outcome that would be observed for unit  $i$  if it were assigned to the treatment (control) condition. Since  $Y_i(1) - Y_i(0)$  represents the treatment effect for unit  $i$ , the SATE is the average of this treatment effect across all units in the sample. But because only one potential outcome can be observed for each unit, we cannot observe the treatment effect for any unit, so the SATE is unknown.

In section 2.4, we learned that randomization of treatment assignment makes the treatment and control groups identical on average. As a result, we can use the *difference-in-means estimator* to estimate the average treatment effect. Let's formalize this argument here. The difference-in-means estimator  $\widehat{\text{SATE}}$  can be written as

$$\begin{aligned} \widehat{\text{SATE}} &= \text{average of the treated} - \text{average of the untreated} \\ &= \frac{1}{n_1} \sum_{i=1}^n T_i Y_i - \frac{1}{n - n_1} \sum_{i=1}^n (1 - T_i) Y_i. \end{aligned} \tag{6.2}$$

Recall that  $n_1$  represents the number of units in the treatment group and hence  $n - n_1$  is the size of the control group. The expression  $\sum_{i=1}^n T_i Y_i$ , for example, gives the sum of the

observed outcome variable across all treated units because the treatment variable  $T_i$  is 1 when unit  $i$  is treated and 0 if it belongs to the control group. This means that  $T_i Y_i = Y_i$  and  $(1 - T_i) Y_i = 0$  when observation  $i$  is in the treatment group, and  $T_i Y_i = 0$  and  $(1 - T_i) Y_i = Y_i$  when it is in the control group.

We now show that the difference-in-means estimator is unbiased for the SATE. As discussed earlier, in survey sampling, the unbiasedness of an estimator means that over repeated sampling the average value of the estimator is identical to the unknown true value of the parameter. In randomized controlled trials, we consider how an estimator behaves over the repeated randomization of treatment assignment. That is, suppose that using a sample of the same  $n$  units, a researcher conducts a randomized controlled trial (infinitely) many times by randomizing the treatment assignment. A given unit will receive the treatment in some of these trials while in others it will be assigned to the control group. Each time, a researcher will compute the difference-in-means estimator after randomizing the treatment assignment and observing the outcome. Throughout the hypothetical repeated experiments, the potential outcomes remain fixed and only the treatment assignment changes. Consequently, unbiasedness implies that the average value of the difference-in-means estimator over repeated trials is equal to the true value of the SATE.

To show the unbiasedness more formally, we can take the expectation of the difference-in-means estimator with respect to  $T_i$  since in this framework the randomized treatment assignment  $T_i$  is the only random variable. Since  $T_i$  is a Bernoulli random variable, its expectation equals  $P(T_i = 1)$ , which is the proportion of subjects who are treated, or  $n_1/n$  in this case:

$$\begin{aligned}\widehat{\mathbb{E}(\text{SATE})} &= \mathbb{E} \left( \frac{1}{n_1} \sum_{i=1}^n T_i Y_i(1) - \frac{1}{n - n_1} \sum_{i=1}^n (1 - T_i) Y_i(0) \right) \\ &= \frac{1}{n_1} \sum_{i=1}^n \mathbb{E}(T_i) Y_i(1) - \frac{1}{n - n_1} \sum_{i=1}^n \mathbb{E}(1 - T_i) Y_i(0) \\ &= \frac{1}{n_1} \sum_{i=1}^n \frac{n_1}{n} Y_i(1) - \frac{1}{n - n_1} \sum_{i=1}^n \left(1 - \frac{n_1}{n}\right) Y_i(0) \\ &= \frac{1}{n} \sum_{i=1}^n \{Y_i(1) - Y_i(0)\} = \text{SATE}. \tag{6.3}\end{aligned}$$

The first equality follows because for a treated unit, the potential outcome under the treatment condition is observed, i.e.,  $Y_i = Y_i(1)$ , while a control unit reveals the other potential outcome, i.e.,  $Y_i = Y_i(0)$ . The second equality holds because the expectation is a linear operator and is taken with respect to the treatment assignment. In other words, the potential outcomes are treated as fixed constants. The derivation above shows that the difference-in-means estimator is unbiased for the SATE.

We can combine the advantage of the preceding random treatment assignment with that of random sampling. Suppose that we first randomly sample  $n$  individuals from a large population of interest. Within this sample, we randomly assign the treatment to  $n_1$  individuals and measure the outcome for each one. This two-step procedure ensures the experimental results are generalizable to the population because the experiment's sample is representative of the population. To see this formally, consider the *population average treatment effect* or PATE,

which represents the average of the treatment effect among all individuals in the population. Here, we use the expectation to represent the population average:

$$\text{PATE} = \mathbb{E}(Y(1) - Y(0)). \quad (6.4)$$

Recall that the sample is representative of the population because of random sampling. This means that while the SATE is unobservable, its expectation equals the PATE. Since the difference-in-means estimator is unbiased for the SATE, the estimator is also unbiased for the PATE. It is also clear from equation (6.3) that the difference-in-means estimator is consistent for the PATE. This result emerges from applying the *law of large numbers* to the sample average of the treatment group and that of the control group, separately. In sum, the combination of random sampling and random assignment enables us to make causal inferences about a target population.

In randomized controlled trials, the average outcome difference between the treatment and control groups is an unbiased estimator of the **sample average treatment effect** (SATE). The estimator is also unbiased and consistent for the **population average treatment effect** (PATE).

A *Monte Carlo simulation* can illustrate the idea of unbiasedness. Suppose that the potential outcome under the control condition  $Y_i(0)$  is distributed according to the *standard normal distribution* in a population (i.e., a normal distribution with zero mean and unit variance). We further assume that in the population the individual-level treatment effect follows another normal distribution with both mean and variance equal to 1. Formally, we can write this hypothetical *data-generating process* as

$$Y_i(0) \sim \mathcal{N}(0, 1) \quad \text{and} \quad Y_i(1) \sim \mathcal{N}(1, 1). \quad (6.5)$$

The treatment assignment is randomized, where a randomly selected half of the sample receives the treatment and the other half does not. Finally, we can define the treatment effect for unit  $i$  as  $\tau_i = Y_i(1) - Y_i(0)$ . For each unit, we observe the potential outcome under the realized treatment condition. Under this model, we can analytically compute the PATE as

$$\mathbb{E}(\tau_i) = \mathbb{E}(Y_i(1)) - \mathbb{E}(Y_i(0)) = 1 - 0 = 1. \quad (6.6)$$

In contrast, the value of the SATE depends on which units are sampled.

Following the reviewed equations, we implement the simulation in Stata to generate one sample of units with potential outcomes from the population. We first create a sample of 100 observations using `set obs`. As before, we use `set seed` so our readers can replicate the same normal distribution and resulting SATE. The `rnormal()` function allows us to draw a random number from a normal distribution with a specified mean (first argument within parentheses) and standard deviation (second argument). Once we calculate the treatment effect at the individual level, we compute the true value of the SATE for our sample. This is, of course, possible only in this hypothetical simulation exercise. In the real world, we will never observe both potential outcomes at the same time for any given observation and hence

the true value of the SATE is unknown. We store the SATE value as a scalar to use in our simulation.

```
. clear all
. set seed 123456
. set obs 100
number of observations (_N) was 0, now 100

.
. * generate a sample
. * Y0 has mean 0 and standard deviation of 1
. generate Y0 = rnormal(0, 1)

. * Y1 has mean 1 and standard deviation of 1
. generate Y1 = rnormal(1, 1)

.
. *individual treatment effect
. generate tau = Y1 - Y0

.
. * true value of the sample average treatment effect
. summarize tau, meanonly
. scalar sate = r(mean)

. display sate
1.0560941
```

We use the `simulate` command to simulate a large number of hypothetical randomized controlled trials by randomly assigning the treatment to the units in the sample and selecting one of the potential outcomes according to the realized treatment condition. In section 5.4.2, we introduced the `program` command that allows us to define a set of calculations that we can repeatedly execute across replications in a Monte Carlo simulation. Here, we define a program called `sate_sim` that, for each replication of the randomized controlled trials, computes the difference-in-means estimator and examines the average performance of this estimator. To randomize the treatment, we use the `runiform()` function to generate a random number from a uniform distribution where all values between 0 and 1 have an equal chance to be drawn. We then sort by this random number and assign the first half of the sample to the treatment and the latter half to the control group. We respectively created two temporary variables `random` and `treat` using the `tempvar` command because the assignment will be regenerated with each simulation. When referencing temporary variables, it is necessary to place the variable name within a grave accent and apostrophe, as we do for local macros. Because we have 100 observations, the entire assignment procedure is equivalent to randomly assigning 50 observations to the treatment group and the other 50 to the control group where we observe  $Y_i(1)$  for the treatment group and  $Y_i(0)$  for the control group. We examine the difference-in-means of the treatment and the control group (`difmeans`), and then the difference between the difference-in-means and the true SATE value (`esterr`). We

store these values as `r()` returns by specifying `rclass` in our program command. In the simulation command, we ask Stata to run 5000 simulations within the `reps` option. After specifying our working directory using `cd`, we save the simulated data set as `sate.dta` (using the `saving` option in the `simulate` command) to plot later. The `nodots` option suppresses Stata's default setting that displays a dot after every simulation.

```
. program sate_sim, rclass
1.           * create temporary assignment numbers and sort
.           tempvar random treat
2.           generate `random' = runiform()
3.           sort `random'
4.           * split sample and assign treatment
.           generate `treat' = cond(_n <= _N / 2, 1, 0)
5.           * calculate treatment means
.           summarize Y0 if `treat' == 0
6.           scalar mu0 = r(mean)
7.           summarize Y1 if `treat' == 1
8.           scalar mu1 = r(mean)
9.           * differences-in-means
.           return scalar difmeans = mu1 - mu0
10.          * estimation error for SATE
.           return scalar esterr = mu1 - mu0 - sate
11. end
```

```
. cd uncertainty
```

```
. simulate esterror = r(esterr) difmeans=r(difmeans), ///
>     saving(sate, replace) reps(5000) nodots: sate_sim
      command: sate_sim
      esterror: r(esterr)
      difmeans: r(difmeans)

. * estimation error for SATE
. summarize esterr

Variable |       Obs        Mean    Std. Dev.         Min         Max
esterror | 5,000  -.0012406   .1635252  -.5323554   .559018
```

We observe that the bias, which is the mean of the estimation error, is  $-0.001$ , close to zero. It is not exactly zero as the theory implies because Monte Carlo simulation adds some noise due to its inherent variability. This deviation from the theoretical value is called *Monte Carlo*

*error.* If we were to conduct this simulation infinitely many times, we could eliminate the Monte Carlo error. In this simulation, the estimation error of the difference-in-means estimator ranges from  $-0.532$  to  $0.559$ . Thus, while the estimator is on average close to the true value of the SATE, it may be far off in any given randomized controlled trial.

To consider the bias of estimating the PATE, we must modify the simulation procedure. Specifically, we add the step of sampling potential outcomes to the program rather than implementing it beforehand; we do this by creating values of  $Y_i(0)$  ( $Y_i(1)$ ) based on a random draw with a mean of  $0$  ( $1$ ) and a standard deviation of  $1$ . We also adjust the program so it can take different values for the mean and standard deviation of  $Y_i(0)$  and  $Y_i(1)$ . This simulates the process where researchers sample individuals from a population and then conduct a randomized experiment. We use `simulate` to repeat this two-step procedure many times. This procedure contrasts with the preceding simulation setting, where we conducted a randomized experiment on the same sample. To compute the new bias, we compare the average value of the difference-in-means estimator over repeated simulations with the true value of PATE, which equals  $1$  in the current example. We specify the type (e.g., integer versus real number) and value of our arguments in the `syntax` command. Variables that are regenerated within our simulation are defined as temporary using `tempvar`. We, again, run 5,000 simulations and ask Stata to suppress the dots.

```

. * PATE simulation
. program pate_sim, rclass
1.      syntax, [obs(integer 100) mu0(real 0) mul(real 1) sigma(real 1)]
2.      drop _all
3.      set obs `obs'
4.      scalar PATE = `mul' - `mu0'
5.      tempvar Y0 Y1 treat
6.      * generate a sample for each simulation: this was previously outside
       the program
7.      generate `Y0' = rnormal(`mu0','sigma')
8.      generate `Y1' = rnormal(`mul','sigma')
9.      * assign treatment condition
10.     generate `treat' = cond(_n <= `obs' / 2, 1, 0)
11.     * calculate and store means for two samples
12.     summarize `Y0' if `treat' == 0
13.     scalar p_mu0 = r(mean)
14.     summarize `Y1' if `treat' == 1
15.     scalar p_mul = r(mean)
16.     * estimation error for PATE
17.     return scalar esterr = p_mul - p_mu0 - PATE
18. end

.
.
.
. simulate esterror = r(esterr), reps(5000) ///
>     nodots: pate_sim, obs(100) mu0(0) mul(1) sigma(1)
      command: pate_sim, obs(100) mu0(0) mul(1) sigma(1)
      esterror: r(esterr)

```

Variable	Obs	Mean	Std. Dev.	Min	Max
esterror	5,000	-.0003001	.2009349	-.749123	.6704996
.					

The average estimation error is close to zero, reflecting the unbiasedness. The variability is greater than in the case of the SATE because random sampling adds more noise.

### 6.1.2 STANDARD ERROR

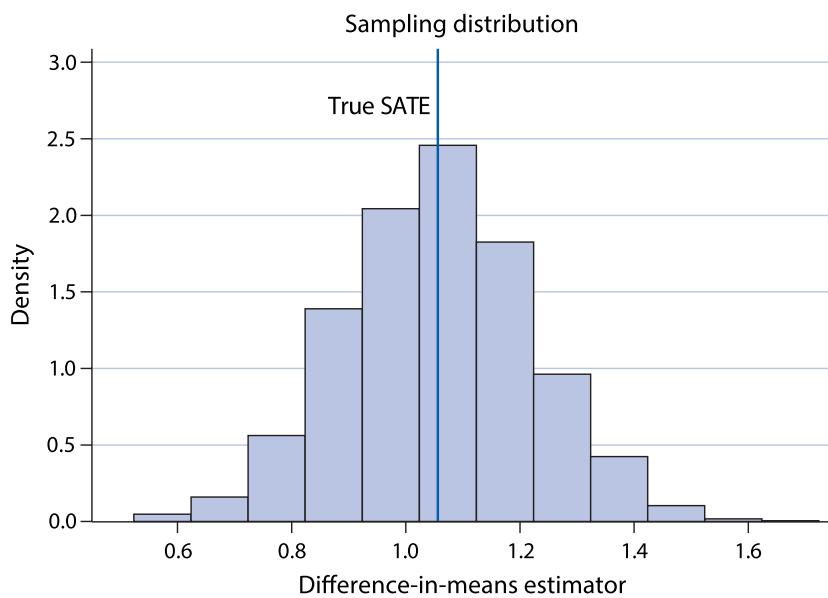
We have focused on the mean of the estimation error, but an unbiased estimator with a large degree of variability is of little use in practice. In our simulation, the difference-in-means estimator was unbiased but its estimation error was sometimes large. We can plot the sampling distribution of the difference-in-means estimator. The histogram shows that while the estimator is accurate on average, it varies significantly from one randomized treatment assignment to another.

```
. use sate, clear
(simulate: sate_sim)

. local x = sate // save sate as local because xline is nonevaluative

.

. histogram difmeans, xline(`x') ///
>         ylabel(0(.5)3) xtitle("Difference-in-means estimator") ///
>         title("Sampling distribution") text(2.8 .95 "True SATE") ///
>         fcolor(none) color(black) width(.1)
(bin=11, start=.52373868, width=.1)
```



How much would an estimator vary over the hypothetically repeated data-generating process? We have used the standard deviation to characterize the spread of distribution in earlier parts of the book, and we can do the same here. This amounts to calculating the standard deviation of the sampling distribution of the difference-in-means estimator.

. summarize difmeans					
Variable	Obs	Mean	Std. Dev.	Min	Max
difmeans	5,000	1.054854	.1635252	.5237387	1.615112

From the `summarize` command, we see that the difference-in-means estimator is on average .164 points away from its mean. This mean equals the true value of the SATE, since the difference-in-means estimator is unbiased for the SATE. Accordingly, the mean of the sampling distribution equals the true value of the SATE, implying that the standard deviation of the sampling distribution (i.e., the deviation from the mean) in this case is equal to the *root-mean-squared-error* (RMSE; i.e., the deviation from the truth) (see section 4.1.3 for the definition of RMSE). In our simulation example, we can compute the RMSE as follows.

```
. generate rmse1 = (difmeans - sate)^2
. summarize rmse1, meanonly
. display sqrt(r(mean))
.16351353
```

The result implies that the estimator is on average .164 points away from the true value of the SATE. A difference between the standard deviation and the RMSE reflects the Monte Carlo error in which the sample average of the estimator differs from its expectation by a small amount.

But if an estimator is biased, then the standard deviation of its sampling distribution will differ from the RMSE. Formally, we can show that the *mean-squared-error* (MSE), which is the square of the RMSE, equals the sum of the variance and squared bias. Let  $\theta$  be a parameter and  $\hat{\theta}$  be its estimator. We can derive this decomposition as follows:

$$\begin{aligned} \text{MSE} &= \mathbb{E}\{(\hat{\theta} - \theta)^2\} \\ &= \mathbb{E}\{(\hat{\theta} - \mathbb{E}(\hat{\theta})) + (\mathbb{E}(\hat{\theta}) - \theta)\}^2 \\ &= \mathbb{E}\{(\hat{\theta} - \mathbb{E}(\hat{\theta}))^2\} + \{\mathbb{E}(\hat{\theta}) - \theta\}^2 \\ &= \text{variance} + \text{bias}^2. \end{aligned}$$

The second equality follows because we simply added and subtracted  $\mathbb{E}(\hat{\theta})$ . The third equality is based on the fact that the cross-product term obtained by expanding the square, i.e.,  $2\mathbb{E}\{(\hat{\theta} - \mathbb{E}(\hat{\theta}))(\mathbb{E}(\hat{\theta}) - \theta)\}$ , can be shown to equal zero.<sup>1</sup>

The decomposition implies that when assessing the accuracy of an estimator, we care about variance as well as bias. An unbiased estimator can have a greater MSE than a biased estimator if the variance of the former is sufficiently larger than that of the latter.

This discussion suggests that we can characterize the variability of an estimator by computing the standard deviation of the *sampling distribution*. Unfortunately, this standard deviation cannot be directly obtained from the data because it is defined over hypothetical repeated random sampling and/or random treatment assignment. In our simulations, we were able to compute it because we generated multiple data sets from the assumed data-generating process. In reality, we obtain only one sample and that is from an unknown data-generating process. However, it turns out that we can estimate the standard deviation of the sampling distribution of an estimator from the observed data. The resulting estimated standard deviation of the sampling distribution is called *standard error* and describes the (estimated) average degree to which an estimator deviates from its expected value.

To characterize the variability of an estimator, we can use the **standard error**, which is an estimated standard deviation of the sampling distribution. One measure of accuracy is the **root-mean-squared error** (RMSE), which measures the average deviation of an estimator  $\hat{\theta}$  from the true parameter value  $\theta$ . The mean-squared error (MSE) of any estimator is equal to the sum of its variance and squared bias:

$$\mathbb{E}\{(\hat{\theta} - \theta)^2\} = \mathbb{V}(\hat{\theta}) + \{\mathbb{E}(\hat{\theta}) - \theta\}^2.$$

As an example, consider the preelection polling described earlier in this chapter. The parameter is the population proportion of voters who support Obama, denoted by  $p$ . We have a simple random sample of  $n$  voters from this population. We can represent each response as an independently and identically distributed (i.i.d.) Bernoulli random variable  $X_i$  with success probability  $p$ , indicating whether respondent  $i$  supports Obama ( $X_i = 1$ ) or not ( $X_i = 0$ ). We use the sample proportion  $\bar{X}_n = \sum_{i=1}^n X_i / n$  as our estimator. Using the rules of variance (see section 5.3.5), we can calculate the variance of this estimator as

$$\mathbb{V}(\bar{X}_n) = \frac{1}{n^2} \mathbb{V}\left(\sum_{i=1}^n X_i\right) = \frac{1}{n^2} \sum_{i=1}^n \mathbb{V}(X_i) = \frac{\mathbb{V}(X)}{n} = \frac{p(1-p)}{n}. \quad (6.7)$$

In this derivation, the second and third equalities are due to the fact that each observation is an i.i.d. random variable. The last equality follows from the fact that the variance of a Bernoulli random variable is  $p(1-p)$ . When  $p$  equals 0.5 (i.e., the population is split into

<sup>1</sup>Specifically, using the rules of expectation, we have  $\mathbb{E}\{(\hat{\theta} - \mathbb{E}(\hat{\theta}))(\mathbb{E}(\hat{\theta}) - \theta)\} = \mathbb{E}[\hat{\theta}\mathbb{E}(\hat{\theta}) - \hat{\theta}\theta - \{\mathbb{E}(\hat{\theta})\}^2 + \mathbb{E}(\hat{\theta})\theta] = \{\mathbb{E}(\hat{\theta})\}^2 - \mathbb{E}(\hat{\theta})\theta - \{\mathbb{E}(\hat{\theta})\}^2 + \mathbb{E}(\hat{\theta})\theta = 0$ .

two exact halves), the standard deviation of the sampling distribution is greatest. Thus, the variance of the estimator is a function of the unknown parameter  $p$ . While we do not know  $p$ , we can estimate it from the observed data. Since, as shown earlier, the sample proportion  $\bar{X}_n$  is an unbiased and consistent estimator of  $p$ , we can use it to construct the following standard error:

$$\text{standard error of sample proportion} = \sqrt{\frac{\bar{X}_n(1 - \bar{X}_n)}{n}}. \quad (6.8)$$

If the sample size is, say, 1,000 and 600 individuals said they supported Obama, then our estimate of Obama's support rate in the population is 0.6 and the standard error is  $.015 \approx \sqrt{0.6(1 - 0.6)/1,000}$ . This implies that our estimate deviates from the true population proportion of Obama supporters by 1.5 percentage points on average.

In general, the standard error must be derived for each statistic because each statistic typically has a unique sampling distribution. For example, the standard error formula given in equation (6.8) applies only to the sample proportion of  $n$  i.i.d. Bernoulli random variables. The derivation in equation (6.7) shows that a more general formula for the standard error of the sample mean, when  $X$  is possibly nonbinary, is given by the following formula.

Suppose that we have a sample of  $n$  independently and identically distributed random variables,  $\{X_1, X_2, \dots, X_n\}$ . The **standard error** of the sample mean  $\bar{X}_n = \sum_{i=1}^n X_i/n$  is given by

$$\text{standard error of sample mean} = \sqrt{\widehat{\mathbb{V}(\bar{X}_n)}} = \sqrt{\frac{\widehat{\mathbb{V}(X)}}{n}}. \quad (6.9)$$

When  $X$  is a Bernoulli random variable, the formula can be simplified as shown in equation (6.8) by setting  $\widehat{\mathbb{V}(X)} = \bar{X}_n(1 - \bar{X}_n)$ .

We can, therefore, compute the standard error by estimating the population variance  $\mathbb{V}(X)$  with the sample variance  $\sum_{i=1}^n (X_i - \bar{X}_n)^2 / (n - 1)$ , where the denominator is  $n - 1$  rather than  $n$  because the estimation of variance requires the estimation of mean, resulting in the loss of one *degree of freedom* (see section 4.3.2).

Finally, we obtain the standard error of the difference-in-means estimator used in a randomized controlled trial. To do this, we note that the variance of the difference-in-means estimator is the sum of the variances of the sample means for the treatment and control groups. We estimate these latter two variances. We can assume statistical independence between the two sample means because they are based on different groups of observations. Our calculations yield the standard error for the difference-in-means estimator when one sample mean is compared with another sample mean.

Suppose that we have a sample of  $n$  independently and identically distributed random variables,  $\{X_1, X_2, \dots, X_n\}$ . We also have another sample of  $m$  independently and identically distributed random variables,  $\{Y_1, Y_2, \dots, Y_m\}$ . Then, the **standard error** of the difference-in-means estimator,  $\sum_{i=1}^n X_i/n - \sum_{i=1}^m Y_i/m$ , is given by

$$\text{standard error of the difference-in-means} = \sqrt{\frac{\widehat{\mathbb{V}}(X)}{n} + \frac{\widehat{\mathbb{V}}(Y)}{m}}. \quad (6.10)$$

We now revisit the simulation conducted at the end of section 6.1.1. In this simulation, the quantity of interest is the PATE rather than the SATE, which is based on a particular sample. We add the standard error calculation to each simulation and obtain 5000 standard errors. These standard errors should on average estimate the standard deviation of the sampling distribution of the difference-in-means estimator. Before defining our new program, we clear our sate data from memory using `clear`. We save the simulated data as `patese.dta`.

```

. clear

. * PATE with standard errors
. program patese, rclass
1.         syntax, [obs(integer 100) mu0(real 0) mul(real 1) sigma(real 1)]
2.         set obs `obs'
3.         scalar PATE = `mul' - `mu0'
4.         * generate randomly assigned treatment groups, as before
5.         tempvar Y0 Y1 treat
6.         generate `Y0' = rnormal(`mu0','sigma')
7.         generate `Y1' = rnormal(`mul','sigma')
8.         generate `treat' = cond(_n <= `obs' / 2, 1, 0)
9.         * store means and variances for two samples
10.        summarize `Y0' if `treat' == 0, detail
11.        scalar p_mu0 = r(mean)
12.        scalar var0 = r(Var)
13.        summarize `Y1' if `treat' == 1, detail
14.        scalar p_mul = r(mean)
15.        scalar var1 = r(Var)
16.        * difference-in-means and standard error
17.        return scalar difmeans = p_mul - p_mu0
18.        return scalar se = sqrt( (var1 / (`obs' / 2)) + (var0 /
19.                               (`obs' / 2)) )
20. end

. simulate difmeans =r(difmeans) se = r(se), ///
>     reps(5000) saving(patese, replace) nodots: patese, ///

```

```
> obs(100) mu0(0) mul(1) sigma(1)
  command: patese, obs(100) mu0(0) mul(1) sigma(1)
  difmeans: r(difmeans)
  se: r(se)

. summarize difmeans se
```

Variable	Obs	Mean	Std. Dev.	Min	Max
difmeans	5,000	.999621	.1997445	.3072847	1.736811
se	5,000	.1994505	.0142112	.1493347	.2525753

As expected from the definition of standard error, the average of the standard errors is close to the standard deviation of the sampling distribution of the difference-in-means estimator. In the current case, we can analytically derive the exact standard deviation of the sampling distribution of this estimator because we know the true data-generating process. Using the distributions of  $Y(1)$  and  $Y(0)$  in equation (6.5), we compute it as

$$\sqrt{\frac{\mathbb{V}(Y(1))}{n_1} + \frac{\mathbb{V}(Y(0))}{n - n_1}} = \sqrt{\frac{1}{50} + \frac{1}{50}} = \frac{1}{5}. \quad (6.11)$$

We see that our simulation procedure approximates this true value very well.

### 6.1.3 CONFIDENCE INTERVALS

To study the properties of an estimator, we have used the mean and standard deviation of its sampling distribution. Next, we consider characterizing the entire sampling distribution rather than its mean and standard deviation. In some special cases, this can be done easily. Let's suppose that  $\{X_1, X_2, \dots, X_n\}$  are independently and identically distributed according to a normal distribution with mean  $\mu$  and variance  $\sigma^2$ . Since the sum of normal random variables follows another normal distribution, the sampling distribution of the sample mean  $\bar{X}_n = \sum_{i=1}^n X_i/n$  is also a normal distribution, with mean  $\mathbb{E}(\bar{X}_n) = \mathbb{E}(X) = \mu$  and variance  $\mathbb{V}(\bar{X}_n) = \sigma^2/n$ .

While the derivation works out nicely in the case of the normal distribution, it is unclear how to characterize the sampling distribution of an estimator in other cases. This is problematic because in practice we do not know the true data-generating process. Fortunately, in many cases of practical interest, there is a way to *approximate* the sampling distribution of an estimator. Specifically, we use the *central limit theorem* introduced in section 5.4.2. The theorem implies that the sampling distribution of the sample mean is approximately normally distributed:

$$\bar{X}_n \stackrel{\text{approx.}}{\sim} \mathcal{N}\left(\mathbb{E}(X), \frac{\mathbb{V}(X)}{n}\right). \quad (6.12)$$

We can derive this result from equation (5.41) of section 5.4.2 by multiplying both sides by the standard deviation,  $\sqrt{\mathbb{V}(X)/n}$  (this changes the variance from 1 to  $\mathbb{V}(X)/n$ ), and adding

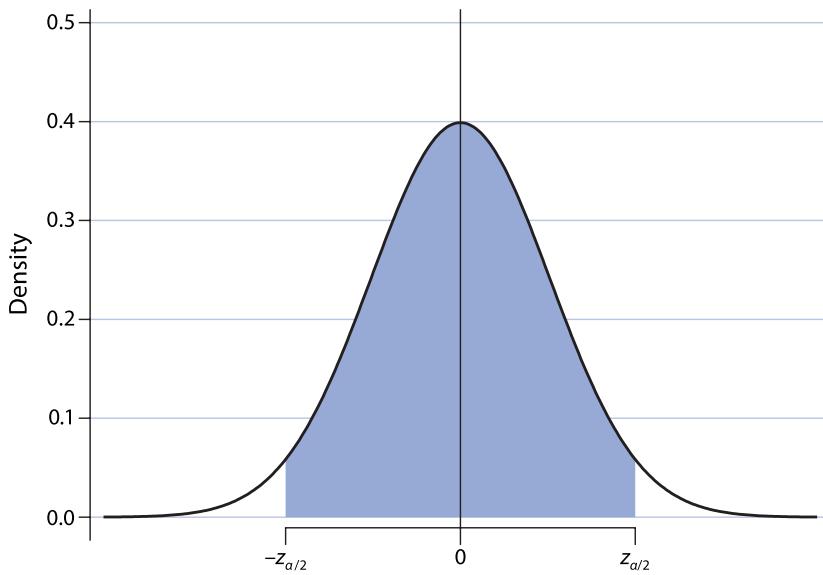


Figure 6.1. Critical Values based on the Standard Normal Distribution. The lower and upper critical values,  $-z_{\alpha/2}$  and  $z_{\alpha/2}$ , are shown on the horizontal axis. The area under the density curve between these critical values (highlighted in blue) equals  $1 - \alpha$ . These critical values are symmetric.

the mean,  $\mathbb{E}(X)$  (this changes the mean from 0 to  $\mathbb{E}(X)$ ). If the random variable is binary, then  $\bar{X}_n$  represents the sample proportion, leaving us with  $\mathbb{E}(X) = p$  and  $\mathbb{V}(X) = p(1 - p)$ . For a large enough sample, this result enables us to characterize the sampling distribution of the sample mean using a normal distribution. Equation (6.12) is useful because it holds regardless of the distribution of the original random variable  $X$ .

Using this result, we can construct another measure of uncertainty called a *confidence interval*. Confidence intervals give a range of values that are likely to include the true value of the parameter. They are also referred to as *confidence bands* or *error bands*. To compute the confidence interval, researchers decide the *confidence level*, or the degree to which they would like to be certain that the interval actually contains the true value. More precisely, over a hypothetically repeated data-generating process, confidence intervals contain the true value of the parameter with the probability specified by the confidence level. Many applied researchers choose the 95% confidence level as a matter of convention but other choices such as 90% and 99% can also be used. The confidence level is often written as  $(1 - \alpha) \times 100\%$ , where  $\alpha$  can take any value between 0 and 1. For example,  $\alpha = 0.05$  corresponds to the 95% confidence level.

Formally, the  $(1 - \alpha) \times 100\%$  asymptotic (i.e., large sample) confidence interval,  $\text{CI}(\alpha)$ , for the sample mean is defined as

$$\text{CI}(\alpha) = [\bar{X}_n - z_{\alpha/2} \times \text{standard error}, \quad \bar{X}_n + z_{\alpha/2} \times \text{standard error}] . \quad (6.13)$$

In this definition,  $z_{\alpha/2}$  is the *critical value*, which equals the  $(1 - \alpha/2)$  quantile of the standard normal distribution such that  $P(Z > z_{\alpha/2}) = 1 - P(Z \leq z_{\alpha/2}) = 1 - \alpha/2$ , where  $Z$  is a standard normal random variable. Thus, the probability that a standard normal random variable is greater than this critical value is equal to  $\alpha/2$ . Figure 6.1 graphically illustrates these critical

**Table 6.1. Commonly Used Critical Values Based on the Normal Distribution for Confidence Intervals.**

$\alpha$	Confidence level %	Critical value $z_{\alpha/2}$	Stata expression
0.01	99	2.58	invnormal(.995)
0.05	95	1.96	invnormal(.975)
0.1	90	1.64	invnormal(.95)

values, where the area under the density curve between the lower and upper critical values, highlighted in blue, equals  $1 - \alpha$ .

The critical values that correspond to commonly chosen confidence levels are shown in table 6.1, along with corresponding Stata expressions. As the confidence level decreases, the critical value decreases, narrowing the width of the confidence interval. This is because the width of the confidence interval is  $2 \times \text{standard error} \times z_{\alpha/2}$  (see equation (6.13)). The trend makes sense because for the same observed data, a shorter confidence interval gives us less confidence that the interval contains the true value. As the table shows, the confidence level corresponds to the argument of the `invnormal()` function. Mathematically, this function computes the inverse of the CDF of a standard normal random variable  $X$ . To use the `invnormal()` function, we input probability  $p$ , and the function returns the quantile  $q$  such that  $p = P(X \leq q)$ . Going back to the survey sampling example, if 600 out of 1,000 respondents support Obama, then the estimate of the population proportion of voters who support Obama, or point estimate, is  $\bar{X}_n = 0.6$  with a standard error of  $.02 = \sqrt{0.6 \times (1 - 0.6) / 1,000}$ . Accordingly, the 99%, 95%, and 90% confidence intervals, with lower and upper bounds, can be computed as follows:

```

. scalar xbar = .6
. scalar se_bar = sqrt(.6 * (1 - .6) / 1000)
. * 99% confidence intervals
. display xbar - invnormal(.995) * se_bar, xbar + invnormal(.995) * se_bar
.56009542 .63990458
. * 95% confidence intervals
. display xbar - invnormal(.975) * se_bar, xbar + invnormal(.975) * se_bar
.56963637 .63036363
. * 90% confidence intervals
. display xbar - invnormal(.95) * se_bar, xbar + invnormal(.95) * se_bar
.57451804 .62548196

```

We observe that a greater confidence level yields a wider confidence interval.

How should we interpret confidence intervals? It is tempting to think, for example, that the probability that the particular 95% confidence interval computed based on the observed

data contains the true value of the parameter is 0.95. However, this interpretation is incorrect. The reason is that the true value of the parameter is unknown and fixed, and hence the probability that a particular confidence interval contains this value is either 1 (when it actually contains it) or 0 (when it does not). We should note that since confidence intervals are a function of observed data, they are random and vary from one (hypothetical) random sample to another. The correct interpretation of confidence intervals is, therefore, that 95% confidence intervals contain the true value of the parameter 95% of the time during a hypothetically repeated data-generating process. In other words, if we had an infinite number of random samples, then 95% of them yield a 95% confidence interval that contains the truth. The probability that a (random) confidence interval includes the true value is called the *coverage probability* (or *coverage rate*). Confidence intervals are valid when they have a coverage probability equal to the nominal value (e.g., 95% in the current example).

We now explain why the confidence interval given in equation (6.13) has a proper coverage rate. Consider the probability that  $(1 - \alpha) \times 100\%$  confidence interval contains the true parameter value  $\mathbb{E}(X)$  (or  $p$  when  $X$  is a Bernoulli random variable), i.e., the probability that the true parameter is between the lower and upper confidence limits. This probability does not change even if we subtract the sample mean  $\bar{X}_n$  from each term and divide it by its standard error:

$$\begin{aligned} & P(\bar{X}_n - z_{\alpha/2} \times \text{standard error} \leq \mathbb{E}(X) \leq \bar{X}_n + z_{\alpha/2} \times \text{standard error}) \\ &= P\left(-z_{\alpha/2} \leq \frac{\mathbb{E}(X) - \bar{X}_n}{\text{standard error}} \leq z_{\alpha/2}\right) \\ &= P\left(-z_{\alpha/2} \leq \frac{\bar{X}_n - \mathbb{E}(X)}{\text{standard error}} \leq z_{\alpha/2}\right) \\ &= 1 - \alpha. \end{aligned} \tag{6.14}$$

The middle probability term,  $\{\mathbb{E}(X) - \bar{X}_n\}/\text{standard error}$ , equals the negative  $z$ -score of the sample mean, which has the same sampling distribution as the  $z$ -score because of symmetry. The central limit theorem implies that the  $z$ -score of the sample mean follows the standard normal distribution when the sample size is sufficiently large:

$$\frac{\bar{X}_n - \mathbb{E}(X)}{\sqrt{\mathbb{V}(X)/n}} \approx \frac{\bar{X}_n - \mathbb{E}(X)}{\text{standard error}} \sim \mathcal{N}(0, 1). \tag{6.15}$$

Therefore, the probability in equation (6.14) equals the blue area of figure 6.1.

We now summarize the standard procedure for constructing asymptotic confidence intervals based on the central limit theorem. The procedure applies to any estimator so long as its asymptotic sampling distribution can be approximated by the normal distribution. Such a normal approximation holds for many cases of interest including almost all the examples in this book.

The **confidence interval** of an estimate  $\hat{\theta}$  can be obtained by using the following procedure:

1. Choose the desired level of confidence  $(1 - \alpha) \times 100\%$  by specifying a value of  $\alpha$  between 0 and 1: the most common choice is  $\alpha = 0.05$ , which gives a 95% confidence level.
2. Derive the sampling distribution of the estimator by computing its mean and variance: in the case of the sample mean, this is given by equation (6.12).
3. Compute the standard error based on this sampling distribution.
4. Compute the critical value  $z_{\alpha/2}$  as the  $(1 - \alpha/2) \times 100$  percentile value of the standard normal distribution: see table 6.1.
5. Compute the lower and upper confidence limits as  $\hat{\theta} - z_{\alpha/2} \times$  standard error and  $\hat{\theta} + z_{\alpha/2} \times$  standard error, respectively.

The resulting confidence interval covers the true parameter value  $\theta$  over a hypothetically repeated data-generating process  $(1 - \alpha) \times 100\%$  of the time.

Several applications of this procedure will be given throughout this section. Here, we conduct additional Monte Carlo simulations to further illustrate the idea of confidence intervals. We first revisit the PATE simulation shown in section 6.1.2. The data should still be in memory. If it is not, it can be opened again by typing `use patese, clear`. Given the estimates and standard errors we computed, we can obtain the 90% and 95% confidence intervals for each of the 5,000 simulations.

```

. * PATE 95% confidence intervals
. generate ci95lo = difmeans - invnormal(.975) * se
. generate ci95hi = difmeans + invnormal(.975) * se
.
. * PATE 90% confidence intervals
. generate ci90lo = difmeans - invnormal(.95) * se
. generate ci90hi = difmeans + invnormal(.95) * se

```

If these confidence intervals are valid, then they should contain the true value of the PATE, which is equal to 1 in this simulation, approximately 95% and 90% of time, respectively. That is exactly what we find below.

```

. * coverage rate for 95% confidence intervals
. generate coverage95 = ci95lo <= 1 & ci95hi >= 1
. * coverage rate for 90% confidence intervals

```

```
. generate coverage90 = ci90lo <= 1 & ci90hi >= 1
.
. summarize coverage95 coverage90
```

Variable	Obs	Mean	Std. Dev.	Min	Max
coverage95	5,000	.949	.2200197	0	1
coverage90	5,000	.8964	.3047714	0	1

As another illustration, we use the polling example described earlier. Again, over repeated random sampling, 95% of the 95% confidence intervals should contain the true parameter value. As the sample size increases, we should observe that the approximation improves with the coverage probability approaching its nominal rate. In the code that follows, we again define a program where we enter the number of observations as one of the parameters and examine, for each simulation, whether the confidence interval contains the truth. We then repeat the simulation using different values of the sample size.

```
. clear
.
. program confint, rclass
  1.      syntax, obs(integer) pr(real) alpha(real)
  2.      clear
  3.      set obs `obs'
  4.      generate d = rbinomial(1, `pr')
  5.      summarize d, meanonly
  6.      scalar xbar = r(mean)
  7.      scalar se = sqrt(xbar * (1 - xbar) / `obs')
  8.      return scalar ciresults = ///
        (`pr' > (xbar - invnormal(1 - `alpha' / 2) * se)) & ///
        (`pr' < (xbar + invnormal(1 - `alpha' / 2) * se))
  9. end
```

```
.
.
. * 50 observations
. simulate ciress50 = r(ciresults), reps(5000) nodots: confint, ///
>     obs(50) pr(.6) alpha(.05)
      command: confint, obs(50) pr(.6) alpha(.05)
      ciress50: r(ciresults)
```

```
. summarize ciress
```

Variable	Obs	Mean	Std. Dev.	Min	Max
ciress50	5,000	.9384	.2404517	0	1

```

. * 100 observations
. simulate ciress50 = r(ciress50), reps(5000) nodots: confint, ///
> obs(100) pr(.6) alpha(.05)
    command: confint, obs(100) pr(.6) alpha(.05)
    ciress100: r(ciress50)

. summarize ciress

      Variable   Obs    Mean    Std. Dev.    Min    Max
ciress100 | 5,000  .9484  .2212402     0      1

. * 1000 observations
. simulate ciress = r(ciress50), reps(5000) nodots: confint, ///
> obs(1000) pr(.6) alpha(.05)
    command: confint, obs(1000) pr(.6) alpha(.05)
    ciress: r(ciress50)

. summarize ciress

      Variable   Obs    Mean    Std. Dev.    Min    Max
ciress | 5,000  .9416  .2345217     0      1

```

The proportion of 95% confidence intervals that contain the true population proportion approaches their nominal value, 95%, as sample size increases.

#### 6.1.4 MARGIN OF ERROR AND SAMPLE SIZE CALCULATION IN POLLS

In the world of polling, the phrase *margin of error* typically refers to the half width of 95% confidence intervals. That is, when we say that Obama's approval rate is 60% with margin of error plus or minus 3 percentage points, we mean that the 95% confidence interval is [57, 63]. In general, we can define the margin of error in polling as

$$\text{margin of error} = \pm z_{0.025} \times \text{standard error} \approx \pm 1.96 \times \sqrt{\frac{\bar{X}_n(1 - \bar{X}_n)}{n}}. \quad (6.16)$$

Now consider the case where the standard deviation of the sampling distribution, i.e.,  $\sqrt{p(1-p)}$ , is the largest. This happens when exactly half of voters support Obama and the other do not, i.e.,  $p = 0.5$ . Then, assuming that we have a large enough sample to ensure  $\bar{X} \approx p$ , the margin of error becomes approximately  $\pm 1/\sqrt{n} \approx \pm 1.96 \times \sqrt{0.5 \times (1 - 0.5)/n}$ . From this result, we can derive the *rule of thumb* commonly applied when researchers are deciding the number of respondents to interview. This rule of thumb states that if you compute the reciprocal of the squared margin of error, it gives the sample size necessary to achieve the

specified level of precision, i.e.,  $n \approx 1/\text{margin of error}^2$ . For example, if we want to obtain an estimate with margin of error plus or minus 3 percentage points, then we need approximately  $1/0.03^2 \approx 1,111$  observations. More generally, by rearranging the terms in equation (6.16), the approximate relationship between the margin of error and sample size can be written as follows.

The **margin of error** of the estimated proportion in polling  $\bar{X}_n$  refers to the half width of the 95% confidence interval or  $z_{0.025} \times \text{standard error} = 1.96 \times \sqrt{\bar{X}_n(1 - \bar{X}_n)/n}$ . The approximate relationship between sample size  $n$  and margin of error is

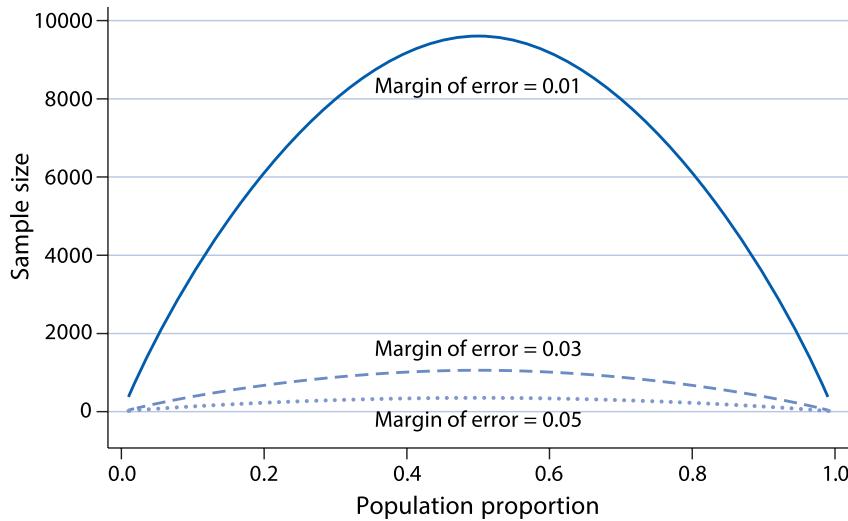
$$n \approx \frac{1.96^2 p(1-p)}{\text{margin of error}^2}, \quad (6.17)$$

where  $p$  is the population proportion. The formula can be used to determine the sample size necessary for conducting a survey.

We can use this formula to determine the sample size for a survey given the desired level of precision, or margin of error, and our prior information about the population proportion  $p$ . This calculation, conducted before fielding a survey, is called the *sample size calculation* and represents an important planning component of survey sampling. We plot the sample size as a function of the population proportion (the horizontal axis) and the margin of error (different line types). In the plot, we observe that a large sample size is required to obtain a margin of error of plus or minus 1 percentage point, particularly when the population proportion is close to 0.5. In contrast, a moderate sample size is sufficient if the desired margin of error is plus or minus 3 percentage points or more.

```
. clear
. set obs 99
number of observations (_N) was 0, now 99
. generate p = _n / 100
. generate n1 = 1.96^2 * p * (1 - p) / (.01^2)
. generate n2 = 1.96^2 * p * (1 - p) / (.03^2)
. generate n3 = 1.96^2 * p * (1 - p) / (.05^2)
.
. line n1 n2 n3 p, xtitle("Population proportion") ///
>           ytitle("Sample size") ///
```

```
> title("Margin of Errors") lpattern(solid dash dot) ///
> legend(label(1 "0.01") label(2 "0.03") label(3 "0.05") col(3))
```



Finally, we revisit the statewide preelection polls analyzed in chapter 4. We analyze the 2008 presidential election polling data set `polls08.dta`, whose variable names and descriptions are given in table 4.2. In section 4.1.3, we computed our estimated margin of victory for Obama within each state and plotted it against his margin of victory on Election Day. We now conduct a similar analysis but include the 95% confidence interval for each estimate and plot it as a vertical line. Note that the official election results are contained in the data file `pres08.dta`, the variable names and descriptions of which are shown in table 4.1. We modify the code that appeared in section 4.1.3 by focusing on Obama's support share and adding the 95% confidence intervals. We assume that the sample size for each poll is 1,000.

```
. use pres08, clear
. merge 1:m state using polls08
      Result                      # of obs.
      not matched                  0
      matched                     1,332  (_merge==3)

. * create a date variable
. generate polldate = date(middate,"YMD")
. format polldate %td
. * compute the number of days to Election Day
```

```

. generate daystoelection = date("2008-11-04", "YMD") - polldate
. * calculate and subset to most recent poll
. bysort state: egen mindays = min(daystoelection)
. keep if daystoelection == mindays
(1,277 observations deleted)

. * calculate and store mean of poll predictions
. collapse (mean) ev obama pollpred = obamapoll, by(state)
.           // convert percent to decimal
.           replace pollpred = pollpred / 100
(51 real changes made)

.           replace obama = obama / 100
(51 real changes made)

. * standard error and confidence intervals
. generate pollpred_se = sqrt(pollpred * (1 - pollpred) / 1000)
. generate pollpred_lo = pollpred - (invnormal(1 - .05 / 2) * pollpred_se)
. generate pollpred_hi = pollpred + (invnormal(1 - .05 / 2) * pollpred_se)

```

We now compare the polling prediction of Obama's support share (the vertical axis) against Obama's vote share on Election Day (the horizontal axis). The idea is that the latter represents the true parameter value within each state. If our 95% confidence intervals are appropriate, 95% of them, which is about 48 states, should contain the actual Election Day result. We use the `twoway rspike` command to draw the confidence interval for each state, with the confidence limits marking each end. We then plot the actual mean value using the `scatter` command.

```

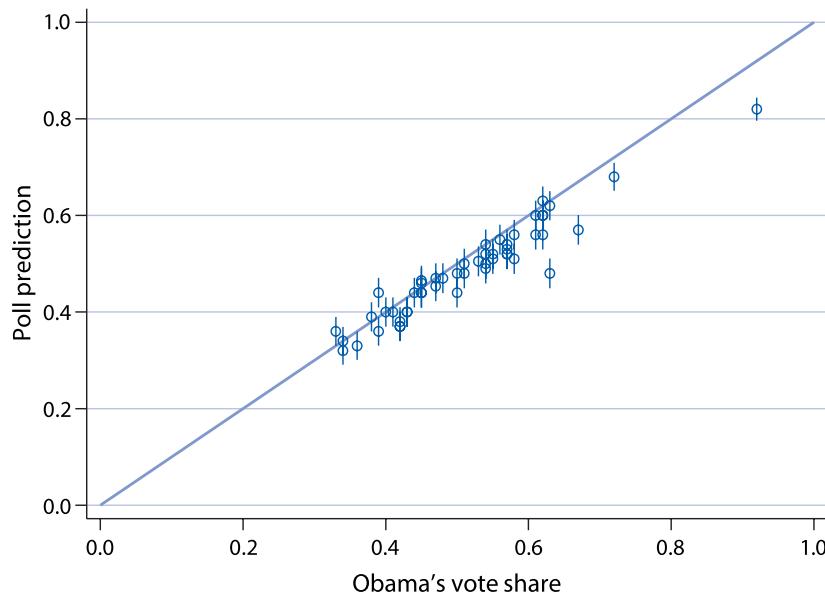
. twoway rspike pollpred_lo pollpred_hi obama || ///
>           scatter pollpred obama, msymbol(Oh) mcolor(black) || ///
>           scatteri 0 0 1 1 , msymbol(none) c(line) ///
>           xtitle("Obama's vote share") ytitle("Poll prediction") ///
>           legend(off)

.

. * proportion of CIs that contain the Election Day outcome
. generate pollcorrect = cond(obama >= pollpred_lo & obama <=
    pollpred_hi, 1, 0)
. summarize pollcorrect

```

Variable	Obs	Mean	Std. Dev.	Min	Max
pollcorrect	51	.5882353	.4970501	0	1



The result suggests that the coverage rate is 58.8%, far below the nominal level. One possible reason for undercoverage is that the poll estimates of Obama's support are biased. If the confidence intervals are not centered around the true parameter value, the coverage rate will be low even if their widths are appropriate. Such bias, if it exists, can affect the confidence intervals by systematically shifting them in one direction and altering the standard error. To investigate this possibility, we first compute the bias and then correct our point estimates by subtracting this bias from the original estimates.

```
. * bias
. generate bias = pollpred - obama
. summarize bias

Variable |   Obs    Mean   Std. Dev.   Min   Max
bias      |    51   -.0267974   .033258   -.15   .05
. * bias corrected estimate
. generate pollbias = pollpred - r(mean)
```

Using this bias-corrected estimate, we can retrospectively compute the standard error and the 95% “bias-corrected” confidence intervals. This retrospective procedure differs from prospective bias correction, which estimates the magnitude of bias without observing the true parameter values (i.e., the Election Day result in this application). We examine the coverage rate of the bias-corrected confidence intervals.

```
. * bias-corrected standard error
. generate bias_se = sqrt(pollbias * (1 - pollbias) / 1000)
. * bias-corrected 95% confidence interval
```

. generate bias_lo = pollbias - (invnormal(1 - .05 / 2) * bias_se)					
. generate bias_hi = pollbias + (invnormal(1 - .05 / 2) * bias_se)					
. * proportion of bias-corrected CIs that contain the Election Day outcome					
. generate biascorrect = cond(obama >= bias_lo & obama <= bias_hi, 1, 0)					
. summarize biascorrect					
Variable	Obs	Mean	Std. Dev.	Min	Max
biascorrect	51	.7647059	.4284033	0	1

The bias correction dramatically improves the coverage rate by almost 20 percentage points. Nevertheless, it is still far from the nominal coverage rate of 95%.

Although the standard errors and confidence intervals represent useful measures of uncertainty, they account only for uncertainty due to random sampling. In practice, other sources of uncertainty remain unaccounted for in the standard error calculation. For example, there may exist systematic bias due to *unit nonresponse*. While beyond the scope of this book, statistical methods have been developed to adjust such bias and underestimation of uncertainty.

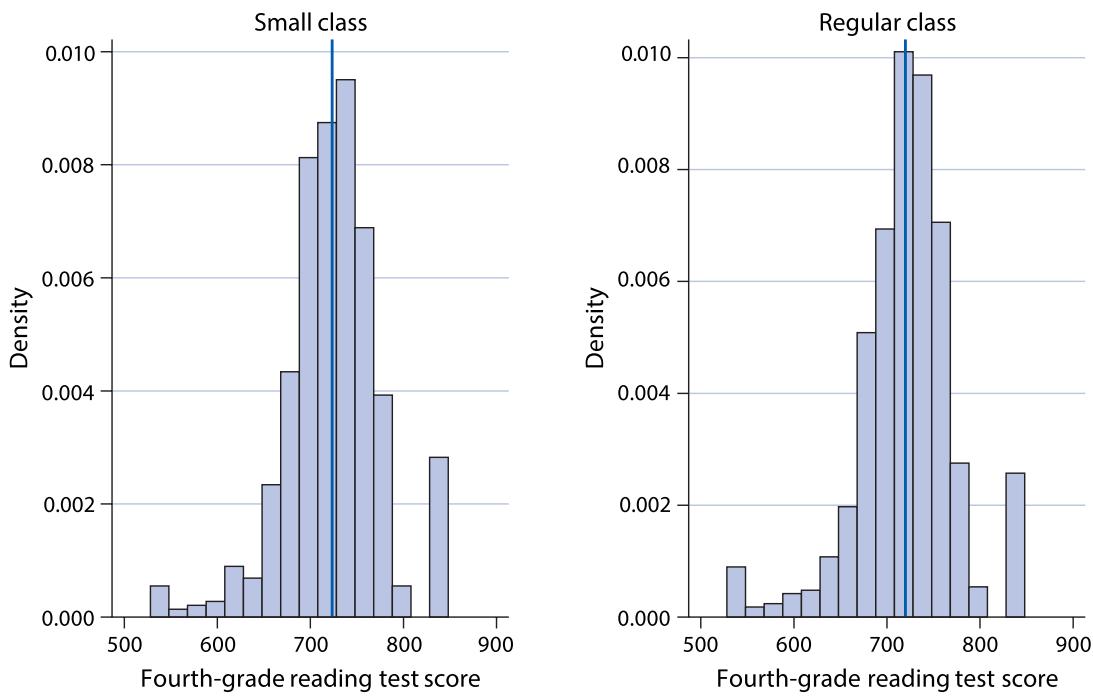
### 6.1.5 ANALYSIS OF RANDOMIZED CONTROLLED TRIALS

We next consider the quantification of uncertainty with respect to estimates of causal effects. We revisit the analysis of data from the STAR (Student–Teacher Achievement Ratio) project introduced in exercise 2.8.1. The STAR project conducted a randomized controlled trial in the 1980s. In the experiment, students were randomly assigned to a small class (`classtype == 1`), regular class (`classtype == 2`), or regular class with an aid (`classtype == 3`). We are interested in knowing whether small class size improves students’ test performance. The data in the file `STAR.dta` have the variable names and descriptions given in table 2.6. We begin by creating a histogram for our outcome variable, the fourth-grade standardized reading test score, separately for students assigned to a regular class and those in a small class. We estimate the average score for each group and add it to each graph as a blue line.

```
. use STAR, clear
. summarize g4reading if classtype == 1, meanonly
. histogram g4reading if classtype == 1, xline(`r(mean)', lcolor(blue)) ///
>           xtitle("Fourth-grade reading test score") title("Small class") ///
>           width(20) fcolor(none) color(black) name(read1, replace)
(bin=16, start=528, width=20)

. summarize g4reading if classtype == 2, meanonly
. histogram g4reading if classtype == 2, xline(`r(mean)', lcolor(blue)) ///
>           xtitle("Fourth-grade reading test score") title("Regular class") ///
>           width(20) fcolor(none) color(black) name(read2, replace)
(bin=16, start=528, width=20)

. graph combine read1 read2
```



We compute the estimated average test score for each treatment group by taking its sample average. These estimates of the average test scores are shown in the above plots using blue lines. We also compute the standard error for each estimator. Since the standard error is the estimated standard deviation of the sampling distribution, it is given by  $\sqrt{\widehat{V}(\bar{X}_n)} = \sqrt{\widehat{\sigma}^2/n}$  in this case. We can use the sample variance as our estimate of the variance parameter  $\sigma^2$ . The `tabstat` command provides these statistics.

```
. * estimate and standard error by class
. tabstat g4reading, by(classtype) statistics(mean semean)

Summary for variables: g4reading
by categories of: classtype
```

classtype	mean	se (mean)
1	723.3912	1.913012
2	719.89	1.83885
3	720.7155	1.864645
Total	721.2478	1.080775

How should one construct a confidence interval for each estimate? As before, we can rely on the central limit theorem and obtain an approximate confidence interval for each estimate. We could perform these calculations, as before, using the `invnormal` function. However, Stata offers the `ci` command, which provides the confidence intervals for means, variances, and

proportions for normally distributed variables. Here, we specify `ci mean` to obtain the confidence intervals for our estimates. The default interval is 95%, but we could use the `level()` option to select a different interval level. Using `return list` after our `ci` command, we can see that Stata stores the statistics as `r()` results. We save the mean and standard error for each class type as scalars for the next calculation. We also save the upper and lower confidence limits, respectively stored in `r(ub)` and `r(lb)`, for later comparison.

```
. * small class size
. ci means g4reading if classtype == 1


| Variable  | Obs | Mean     | Std. Err. | [95% Conf. Interval] |
|-----------|-----|----------|-----------|----------------------|
| g4reading | 726 | 723.3912 | 1.913012  | 719.6355 727.1469    |


. return list

scalars:
      r(N) = 726
      r(mean) = 723.3911845730028
      r(se) = 1.913012295245824
      r(lb) = 719.6354795228319
      r(ub) = 727.1468896231736
      r(level) = 95

macros:
      r(citype) : "normal"

. scalar est_small = r(mean)
. scalar se_small = r(se)
. scalar ciло_small = r(lb)
. scalar cihi_small = r(ub)
.
. * regular class size
. ci means g4reading if classtype == 2


| Variable  | Obs | Mean   | Std. Err. | [95% Conf. Interval] |
|-----------|-----|--------|-----------|----------------------|
| g4reading | 836 | 719.89 | 1.83885   | 716.2806 723.4993    |


. scalar est_reg= r(mean)
. scalar se_reg = r(se)
. scalar ciло_reg = r(lb)
. scalar cihi_reg = r(ub)
```

The results show that the confidence intervals overlap with each other. Does this mean that the estimated average difference between the two groups, or the estimated PATE of small class size, is not statistically significant? An estimated effect is statistically significant if it

reflects true patterns in the population, rather than arising from mere chance. To find out the answer to this question, it would be best to compute the confidence interval directly for the estimated average difference. Recall the standard error of the difference-in-means estimator given in equation (6.10). Using this standard error formula, we can compute the 95% confidence interval for the estimated PATE.

```
. * difference-in-means estimator
. scalar ate_est = est_small - est_reg
. display ate_est
3.5012324

. * standard error and 95% confidence interval
. scalar ate_se = sqrt(se_small^2 + se_reg^2)
. display ate_se
2.6534853

. scalar ate_lo = ate_est - invnormal(1 - .05 / 2) * ate_se
. scalar ate_hi = ate_est + invnormal(1 - .05 / 2) * ate_se
. display ate_lo, ate_hi
-1.6995032 8.701968
```

We find that the average treatment effect of small class size on the fourth grade reading score is estimated to be 3.50 with a standard error of 2.65. The 95% confidence interval is  $[-1.70, 8.70]$ , containing zero. This finding suggests that although the estimated average treatment effect is positive, it features a considerable degree of uncertainty.

### 6.1.6 ANALYSIS BASED ON STUDENT'S $t$ -DISTRIBUTION

The calculation of confidence intervals has so far relied upon the central limit theorem. This is why we used the quantiles of the standard normal distribution when computing confidence intervals, assuming that we have a large enough sample to invoke the central limit theorem. This assumption is useful because the central limit theorem applies to a wide variety of distributions. Given that we often do not know the distribution of an outcome variable, the procedure of constructing confidence intervals described earlier is quite general.

We consider an alternative assumption, that the outcome variable (rather than its sample mean) is generated from a normal distribution. As an illustration, we apply this assumption to the STAR experiment just analyzed in section 6.1.5. We assume that the test scores for each group follow a normal distribution, with possibly different means and variances. While the histograms shown earlier suggest that the distribution of test scores for each group may not satisfy this assumption, the inference resulting from this assumption proves more conservative than the asymptotic inference we have been using based on the central limit theorem. Because many researchers prefer conservative inferences, they often use confidence intervals under this normally distributed outcome assumption even when the assumption is not justifiable.

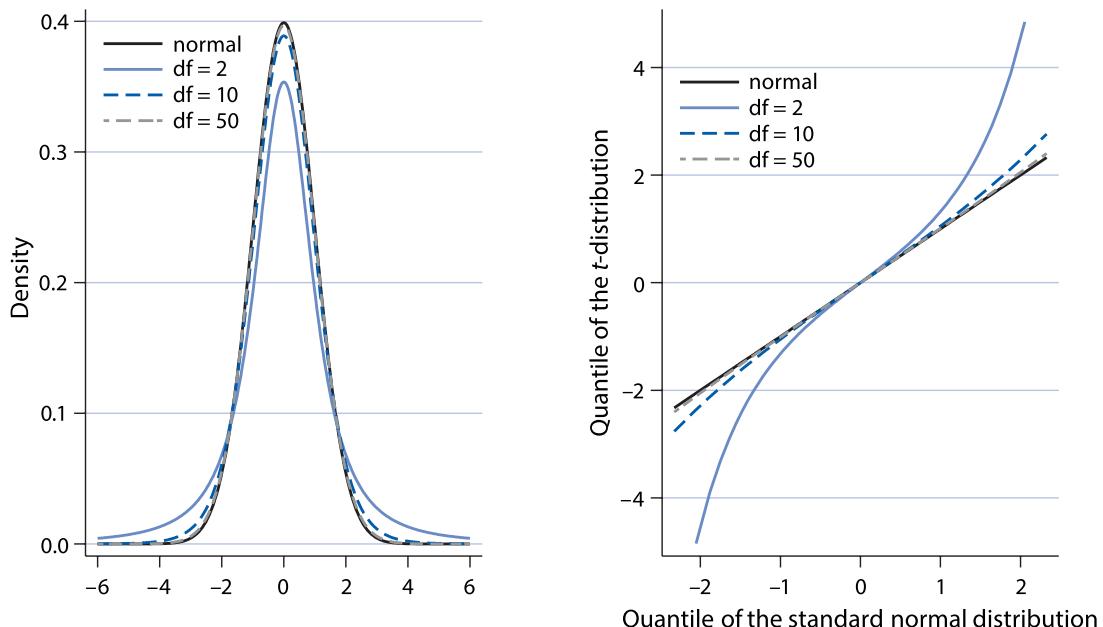
When a random variable is normally distributed, we can obtain an exact confidence interval for the sample mean using *Student's  $t$ -distribution*, also simply called the  *$t$ -distribution*.

The name of the distribution originates from the fact that its British creator William Gossett, a researcher at beer producer Guinness, published the paper introducing it under the pseudonym “Student.” We use  $t_v$  to represent the  $t$ -distribution with  $v$  degrees of freedom. Specifically, the  $z$ -score of the sample mean is called the  **$t$ -statistic** and is distributed according to the  $t$ -distribution with  $n - 1$  degrees of freedom. Roughly, the degrees of freedom represent the number of independent observations used for estimation minus the number of parameters to be estimated (see section 4.3.2). The current case involves one parameter to estimate: we use the standard error to estimate the standard deviation of the sampling distribution. This result holds exactly so we do not resort to asymptotic approximation.

Suppose that  $\{X_1, X_2, \dots, X_n\}$  are  $n$  independently and identically distributed random variables from a normal distribution with mean  $\mu$  and variance  $\sigma^2$ . Then, the  $z$ -score of the sample mean  $\bar{X}_n$ , which is called the  **$t$ -statistic**, follows **Student's  $t$ -distribution** with  $n - 1$  degrees of freedom:

$$\text{ $t$ -statistic of sample mean} = \frac{\bar{X}_n - \text{mean}}{\text{standard error}} = \frac{\bar{X}_n - \mu}{\hat{\sigma}} \sim t_{n-1}.$$

The  $t$ -distribution is quite similar to the standard normal distribution but has heavier tails. In the left-hand plot that follows, we graphically compare the density function of the  $t$ -distribution with 3 different degrees of freedom to the standard normal distribution. The  $t$ -distribution with  $v$  degrees of freedom has mean zero. The variance is given by  $v/(v - 2)$  when the number of degrees of freedom is greater than 2. It turns out that the variance does not exist when  $v$  is less than or equal to 2.



Like the standard normal distribution (black solid line), the  $t$ -distribution density function is symmetric and centered around zero. But the  $t$ -distribution has more mass in the tail areas than the standard normal distribution, especially when the degrees of freedom are small. As the degrees of freedom increase, the  $t$ -distribution approaches the standard normal distribution. This makes sense because according to the *central limit theorem*, the  $z$ -score of the sample mean follows the standard normal distribution regardless of the distribution of the original random variable. Therefore, the standard normal distribution should approximate the sampling distribution of the  $t$ -statistic well for a sufficiently large sample size. The construction of the confidence intervals under this setting is the same as for the sample mean, except that we use the  $t$ -distribution with  $n - 1$  degrees of freedom when computing the *critical values*. That is, the  $(1 - \alpha) \times 100\%$  confidence interval for the sample mean is given by equation (6.13) except that  $z_{\alpha/2}$  now equals the  $1 - \alpha/2$  quantile of the  $t$ -distribution. This results in a wider, and hence more conservative, confidence interval because the critical values based on the  $t$ -distribution are greater than those based on the standard normal distribution (see the *quantile-quantile plot* or *Q-Q plot* in the right-hand panel of the previous figure). For example, when  $\alpha = 0.05$  and  $n = 50$ , the critical value based on the  $t$ -distribution is equal to 2.01, which is slightly greater than the one based on the standard normal distribution, 1.96.

We now go back to the analysis of the STAR data and compute the 95% confidence intervals for the estimated average reading score under each treatment condition. To compute critical values for  $t$ -statistics in Stata, we could use the `invt(df, p)` function with the `df` argument specifying the degrees of freedom and the `p` argument specifying  $z_{\alpha/2}$ , rather than the `invnormal()` function we used for the normal approximation. A more straightforward option, though, is to use Stata's `ttest` command, confining our analysis to just small and regular classes. We include the `unequal` option because we do not assume equal variance across the two class types. As we will review later in this chapter, the `ttest` command provides us with a comparison of means, but it also displays the 95% confidence intervals for each class size (using the `by` option), along with the degrees of freedom. We will use the `ttest` command later for hypothesis testing, but we now focus on just the last two columns of the output.

```
. * 95% confidence intervals, by classtype
. ttest g4reading if classtype == 1 | classtype == 2 , by(classtype) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
1	726	723.3912	1.913012	51.54494	719.6355 727.1469
2	836	719.89	1.83885	53.16788	716.2806 723.4993
combined	1,562	721.5173	1.326654	52.4322	718.9151 724.1195
diff		3.501232	2.653485		-1.703591 8.706055

```

diff = mean(1) - mean(2)                                t = 1.3195
Ho: diff = 0                                         Satterthwaite's degrees of freedom = 1541.25
Ha: diff < 0                                         Ha: diff != 0
Pr(T < t) = 0.9064                                     Pr(|T| > |t|) = 0.1872
                                                               Ha: diff > 0
                                                               Pr(T > t) = 0.0936

.
. * 95% CI based on the central limit theorem
. display ci.lo_small, ci.hi_small // small class size
719.63548 727.14689

. display ci.lo_reg, ci.hi_reg // regular class size
716.28064 723.49926

```

We see that the confidence intervals are similar to those obtained using the central limit theorem. The differences are tiny because the sample size is relatively large. To compute the confidence interval for the difference-in-means estimator, suppose that  $\{X_1, X_2, \dots, X_n\}$  are  $n$  independently and identically distributed normal random variables with mean  $\mu_X$  and variance  $\sigma_X^2$ , and  $\{Y_1, Y_2, \dots, Y_m\}$  are  $m$  i.i.d. normal random variables with mean  $\mu_Y$  and variance  $\sigma_Y^2$ . Then, the  $t$ -statistic is given by

$$t\text{-statistic of difference-in-means} = \frac{(\bar{X}_n - \bar{Y}_m) - (\mu_X - \mu_Y)}{\sqrt{\hat{\sigma}_X^2/n + \hat{\sigma}_Y^2/m}}. \quad (6.18)$$

Although this  $t$ -statistic also follows Student's  $t$ -distribution, the degrees of freedom calculation is complicated. The details of this calculation are beyond the scope of this book, but we can construct the confidence interval based on Student's  $t$ -distribution. Stata provides the degrees of freedom in the `ttest` output. The degrees of freedom are calculated as 1,541.2. Because the size of our sample is not too small, the resulting confidence interval is close to the one based on the normal approximation.

## 6.2 Hypothesis Testing

In section 5.1.5, we presented an analysis of Arnold Schwarzenegger's 2009 veto message to the California legislature, and showed that the particular order of words in his message was highly unlikely to be a consequence of coincidence alone. This was done by examining the likelihood of observing the event that actually happened under a particular probability model. In this section, we formalize this logic and introduce a general principle of statistical *hypothesis testing* that underlies such analysis. This principle enables us to determine whether or not the occurrence of an observed event is likely to be due to chance alone.

### 6.2.1 TEA-TASTING EXPERIMENT

In his classic book *The Design of Experiments*, Ronald Fisher introduced the idea of a statistical hypothesis test. During an afternoon tea party at the University of Cambridge, a lady declared that tea tastes different depending on whether the tea is poured into the milk or the milk is poured into the tea. Fisher examined this claim by using a randomized experiment

**Table 6.2.** M and T represent two scenarios, “milk is poured first” and “tea is poured first,” respectively. Under the hypothesis that the lady has no ability to distinguish the order in which milk and tea were poured into each cup, her guess will be identical regardless of which cups had milk/tea poured first.

Cups	Lady's guess	Actual order	Scenarios			...
1	M	M	T	T	T	
2	T	T	T	T	M	
3	T	T	T	T	M	
4	M	M	T	M	M	
5	M	M	M	M	T	
6	T	T	M	M	T	
7	T	T	M	T	M	
8	M	M	M	M	T	
Number of correct guesses		8	4	6	2	...

in which eight identical cups were prepared and four were randomly selected for milk to be poured into the tea. For the remaining four cups, the milk was poured first. The lady was then asked to identify, for each cup, whether the tea or the milk had been poured first. To everyone’s surprise, the lady correctly classified all the cups. Did this happen by luck or did the lady actually possess the ability to detect the order, as she claimed?

To analyze this randomized experiment, we draw on potential outcomes as explained in chapter 2. For each of the eight cups, we consider two potential guesses given by the lady, which may or may not depend on whether milk or tea was actually poured into the cup first. If we hypothesize that the lady had no ability to distinguish whether milk or tea was poured into the cup first, then her guess should not depend on the actual order in which milk and tea were poured. In other words, under this hypothesis, the two potential outcomes should be identical. Recall the *fundamental problem of causal inference*, which states that only one of the two potential outcomes can be observed. Here, the hypothesis that the lady possesses no ability to distinguish the two types of tea with milk reveals her responses under counterfactual scenarios.

Fisher’s analysis proceeds under this hypothesis and involves computing the number of correctly guessed cups under every possible assignment combination. As discussed in section 6.1.1, this experiment is an example of *complete randomization*, where the number of observations assigned to each condition is fixed a priori. In contrast, *simple randomization* would randomize each cup independently without such a constraint. Table 6.2 illustrates Fisher’s method. The second column of the table shows the lady’s actual guess for each cup, which is identical to the true order (third column) in which milk and tea were poured into the cup. In the remaining columns, we show three arbitrarily selected combinations of assigning four cups to “milk first” and the other four to “tea first.” Although these counterfactual assignment combinations did not occur in the actual experiment, we can compute the number of correctly guessed cups under each scenario with the aforementioned hypothesis that the lady lacks the ability to distinguish between the two types of tea with milk and thus different

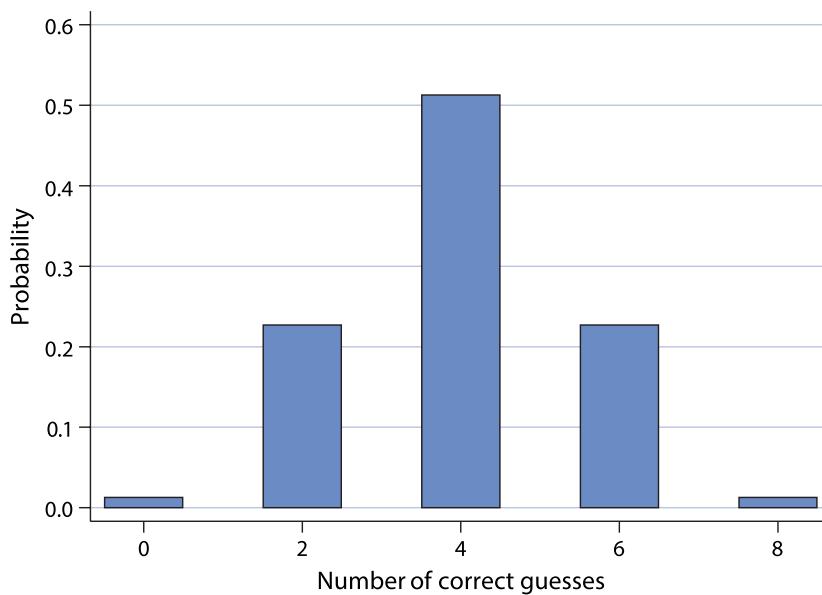


Figure 6.2. Sampling Distribution for the Tea-Tasting Experiment. The barplot shows the distribution of the number of correctly classified cups.

assignments do not affect the lady's guess. This is done by simply comparing the lady's guess (second column), which is assumed to remain unchanged, with each counterfactual assignment. For example, if the cups had received the assignments in the fifth column of the table, then the number of correctly classified cups would have been 6. Under this setup, the key question concerns the likelihood that the lady would have classified all 8 cups correctly if she had not had the ability to distinguish the taste difference. Since each assignment combination is equally likely in this randomized experiment, we can compute the probability of perfect classification by counting the number of ways in which we assign four cups to the "milk first" condition and the remaining four cups to the "tea first" condition (see equation (5.1)). The number of combinations is given by  ${}_8C_4 = 8!/(4! \times (8 - 4)!) = 70$  because four cups out of eight were selected to have tea poured in first. Accordingly, under the assumption that the lady has no ability to distinguish the taste difference, the probability that she guesses all cups correctly is  $1/70$ , or approximately 0.01, which is quite small. We conclude from this analysis that the lady's perfect classification is unlikely to have occurred due to chance alone.

Moreover, as shown in figure 6.2, we can characterize the exact distribution of the number of correctly specified cups over all possible assignment combinations. How is this distribution derived? First, there is only one assignment combination, presented as the actual order in the third column of the table, that makes the lady's guesses a set of perfect classifications. Similarly, there is one assignment combination that makes all of her guesses incorrect. In this experiment, the number of ways in which the lady guesses two cups correctly is equivalent to the product of two things: the number of ways in which the lady correctly classifies one of the four "milk first" conditions and the number of ways in which the lady incorrectly classifies three of them. We can compute this as  ${}_4C_1 \times {}_4C_3 = 16$ . The same calculation applies to the number of assignment combinations that leads to six correctly classified cups. Similarly, we can compute the number of combinations that lead to four correctly classified cups, which is given by  ${}_4C_2 \times {}_4C_2 = 36$ . Finally, because by design the number of cups assigned to

each condition is equal, there is no instance where the number of correctly classified cups is odd. Below, we compute the probability of each event by using the `comb()` function, which enables us to compute combinations.

```
. clear
. set obs 5
number of observations (_N) was 0, now 5
. * use number of correctly classified cups as identifier
. egen guess = fill(0 2 4 6 8)
.
. * truth: enumerate the number of assignment combinations
. generate true = comb(4,0) * comb(4,4) if guess == 0
(4 missing values generated)

. replace true = comb(4,1) * comb(4,3) if guess == 2
(1 real change made)

. replace true = comb(4,2) * comb(4,2) if guess == 4
(1 real change made)

. replace true = comb(4,3) * comb(4,1) if guess == 6
(1 real change made)

. replace true = comb(4,4) * comb(4,0) if guess == 8
(1 real change made)

. list true



|    | true |
|----|------|
| 1. | 1    |
| 2. | 16   |
| 3. | 36   |
| 4. | 16   |
| 5. | 1    |



.
. * compute probability: divide it by the total number of events
. summarize true

    Variable |       Obs        Mean     Std. Dev.      Min      Max
-------------|-----|-----|-----|-----|-----|
          true |       5        14     14.40486         1        36
.
. generate trueprob = true / r(sum)
```

```
. list guess trueprob
```

	guess	trueprob
1.	0	.0142857
2.	2	.2285714
3.	4	.5142857
4.	6	.2285714
5.	8	.0142857

Once again, we can also approximate this distribution using Monte Carlo simulations. We generate 1,000 hypothetical experiments to approximate the sampling distribution of the number of correctly classified cups. To do this, we use the `simulate` command to repeat a program that randomly assigns four cups to the “milk first” (M) condition and the remaining four to the “tea first” (T) condition. We then compute the fraction of trials that yield a certain number of correctly specified cups. The following code block shows this simulation approach. We find that the differences between the simulation results and the analytical answers are quite small.

```
. set seed 12345
. program ladytea, rclass
1.         syntax
2.         clear
3.         set obs 8
4.         * create temporary variables that will be regenerated in
           simulation
.         tempvar unif correct ladyguess sampleguess
5.         * assign first half of observations "milk first," second half
           "tea first"
.         generate `sampleguess' = cond(_n <= _N / 2, "M", "T")
6.         * generate random number and re-sort order/rows
.         generate `unif' = runiform()
7.         sort `unif'
8.         * if re-sorted row equals 1, 4, 5, 8, assign ladyguess M,
           assign T otherwise
.         generate `ladyguess' = cond(inlist(_n, 1, 4, 5, 8), "M", "T")
9.         * mark as correct if sample guess equals lady guess, save
           result
.         generate `correct' = cond(`sampleguess' == `ladyguess', 1, 0)
10.        summarize `correct'
```

```

11.           return scalar correct = r(sum)
12. end

.
. preserve

.   simulate guess = r(correct), reps(1000) nodots: ladytea
      command: ladytea
      guess: r(correct)

.   tabulate guess



| r(correct) | Freq. | Percent | Cum.   |
|------------|-------|---------|--------|
| 0          | 18    | 1.80    | 1.80   |
| 2          | 247   | 24.70   | 26.50  |
| 4          | 496   | 49.60   | 76.10  |
| 6          | 223   | 22.30   | 98.40  |
| 8          | 16    | 1.60    | 100.00 |
| Total      | 1,000 | 100.00  |        |



. generate correct = 1
. collapse (sum) correct, by(guess)
. tempfile sim
. save sim, replace
(note: file sim.dta not found)
file sim.dta saved

```

```

. restore

.

. * estimated probability for each number of correct guesses
. merge 1:1 guess using sim, keep(3)



| Result      | # of obs.     |
|-------------|---------------|
| not matched | 0             |
| matched     | 5 (_merge==3) |



. generate correctpct = correct / 1000
. * comparison with analytical answers; the differences are small
. generate dif = correct / 1000 - trueprob
. list guess correctpct dif

```

	guess	corr_pct	dif
1.	0	.018	.0037143
2.	2	.247	.0184286
3.	4	.496	-.0182857
4.	6	.223	-.0055714
5.	8	.016	.0017143

```
. erase sim.dta
```

The major advantage of Fisher's analysis is that the inference is based solely on the randomization of treatment assignment. Such inference is called *randomization inference*. Methods based on randomization inference typically do not require a strong assumption about the data-generating process because researchers control the randomization of treatment assignment, which alone serves as the basis of inference.

### 6.2.2 THE GENERAL FRAMEWORK

The tea-tasting experiment illustrates a general framework called statistical hypothesis testing. Statistical hypothesis testing is based on probabilistic *proof by contradiction*. Proof by contradiction is a general strategy of mathematical proof in which one demonstrates that assuming the contrary of what we would like to prove leads to a logical contradiction. Consider the proposition that there is no smallest positive *rational number*. To prove this proposition, we assume that the conclusion is false. Suppose that there exists a smallest positive rational number  $a$ . Recall that any rational number can be expressed as the fraction of two integers:  $a = p/q > 0$ , where both the numerator  $p$  and the nonzero denominator  $q$  are positive integers. But, for example,  $b = a/2$  is smaller than  $a$ , and yet  $b$  is also a rational number. This contradicts the hypothesis that  $a$  is the smallest positive rational number.

In the case of statistical hypothesis testing, we can never reject a hypothesis with 100% certainty. Consequently, we use a probabilistic version of proof by contradiction. We begin by assuming a hypothesis we would like to eventually refute. This hypothesis is called a *null hypothesis*, often denoted by  $H_0$ . In the current application, the null hypothesis is that the lady has no ability to tell whether milk or tea is poured first into a cup. This is an example of a *sharp null hypothesis* because all potential outcomes for each observation are determined, and therefore known, under this hypothesis. In contrast, we will later consider a *nonsharp null hypothesis*, which fixes the *average* potential outcome rather than every potential outcome.

Second, we choose a *test statistic*, which is some function of observed data. For the tea-tasting experiment, the test statistic is the number of correctly specified cups. Next, under the null hypothesis, we derive the *sampling distribution* of the test statistic, which is given in figure 6.2 for our application. This distribution is also called the *reference distribution*. Finally, we ask whether the observed value of the test statistic is likely to occur under the reference distribution. In the current experiment, the number of correctly classified cups is observed

**Table 6.3.** Type I and Type II Errors in Hypothesis Testing.

	Reject $H_0$	Retain $H_0$
$H_0$ is true	<b>type I error</b>	correct
$H_0$ is false	correct	<b>type II error</b>

$H_0$  represents the null hypothesis.

to be 8. If 8 is likely under the reference distribution, we retain the null hypothesis. If it is unlikely, then we reject the null hypothesis.

In this textbook, we prefer to use phrases such as “fail to reject the null hypothesis” and “retain the null hypothesis” instead of “accept the null hypothesis.” Philosophical views on this issue differ, but we adopt a perspective that failure to reject the null hypothesis is evidence for some degree of consistency between the data and the hypothesis, but does not necessarily indicate the correctness of the null hypothesis. Others, however, argue that the failure to reject the null hypothesis implies acceptance of the hypothesis. Regardless of one’s stance on this issue, statistical hypothesis testing provides empirical support for scientific theories.

How should we quantify the degree to which the observed value of the test statistic is unlikely to occur under the null hypothesis? We use the *p-value* for this purpose. The *p-value* can be understood as the probability that under the null hypothesis, we observe a value of the test statistic at least as extreme as the one we actually observed. A smaller *p-value* provides stronger evidence against the null hypothesis. Importantly, the *p-value* does not represent the probability that the null hypothesis is true. This probability is actually either 1 or 0 because the null hypothesis is either true or false, though researchers do not know which.

To decide whether or not to reject the null hypothesis, we must specify the *level of test*  $\alpha$  (as explained later, this  $\alpha$  is the same as the confidence level  $\alpha$  for confidence intervals discussed earlier). If the *p-value* is less than or equal to  $\alpha$ , then we reject the null hypothesis. The level of test represents the probability of false rejection if the null hypothesis is true. This error is called *type I error*. Typically, we would like the level of test to be low. Commonly used values of  $\alpha$  are 0.05 and 0.01.

Table 6.3 shows two types of errors in hypothesis testing. While researchers can specify the degree of type I error by choosing the level of test  $\alpha$ , it is not possible to directly control *type II error*, which results when researchers retain a false null hypothesis. Notably, there is a clear trade-off between type I and type II errors in that minimizing type I error usually increases the risk of type II error. As an extreme example, suppose that we never reject the null hypothesis. Under this scenario, the probability of type I error is 0 if the null hypothesis is true, but the probability of type II error is 1 if the null hypothesis is false.

In the case of the tea-tasting experiment, the test statistic is the number of correctly classified cups. Since the observed value of this test statistic was 8, which is the most extreme value, the *p-value* equals the probability that the number of correct guesses is 8 or  $1/70 \approx 0.014$ . If the lady correctly classified six cups instead of eight, two values are at least as extreme as the observed value: 6 and 8. Therefore, in this case, the *p-value* is  $({}_4C_0 \times {}_4C_4 + {}_4C_1 \times {}_4C_3)/70 = (1 + 16)/70 \approx 0.243$ .

These  $p$ -values are *one-sided p-values* (or *one-tailed p-values*) because they consider only the values of the test statistic that are greater than or equal to the observed value. Under this one-sided *alternative hypothesis*, which is the complement of the null hypothesis, we ignore an extreme response on the other side, such as classifying all eight cups incorrectly. In contrast, if we specify a two-sided alternative hypothesis, then computing the *two-sided p-value* (or *two-tailed p-value*) requires consideration of extreme values on both sides. If the reference distribution is symmetric, then the two-sided  $p$ -value is twice as great as the one-sided value. In the tea-tasting experiment, the two-sided  $p$ -value is  $2/70 \approx 0.029$ . If the lady had correctly guessed six cups, then the two-sided  $p$ -value is  $2 \times (1 + 16)/70 \approx 0.486$ .

While the framework described here is applicable to any statistical hypothesis testing, the particular hypothesis testing procedure used for the tea-tasting experiment is called *Fisher's exact test*. As explained earlier, this test is an example of randomization inference, where the validity of the test can be justified based on the randomization of treatment assignment.

Fisher's exact test can be implemented in Stata with the `exact` option in the `tabulate` command, producing a  $2 \times 2$  contingency table, where rows and columns represent a binary treatment assignment variable and a binary outcome variable, respectively. As examples, we create tables for the tea-tasting experiment: one case with all 8 cups correctly classified and the other case with six out of eight cups correctly classified. In each table, rows represent actual assignments and columns provide reported guesses with the diagonal elements corresponding to the correct guesses. By default, the `exact` option will return both the one-sided and the two-sided test results. In the following code, we conduct the Fisher's exact test for both eight correct guesses and for six correct guesses.

```
. clear
. set obs 8
number of observations (_N) was 0, now 8
. generate treat = cond(inlist(_n, 1, 4, 5, 8), 1, 0)
. generate guess8 = treat
. generate guess6 = cond(inlist(_n, 1, 3, 5, 8), 1, 0)
. * one-sided test for 8 correct guesses (1-sided Fisher's exact)
. tabulate treat guess8, exact



| treat | guess8 |   | Total |
|-------|--------|---|-------|
|       | 0      | 1 |       |
| 0     | 4      | 0 | 4     |
| 1     | 0      | 4 | 4     |
| Total | 4      | 4 | 8     |


Fisher's exact = 0.029
1-sided Fisher's exact = 0.014
```

- . \* one-sided test for 6 correct guesses (Fisher's exact)
- . tabulate treat guess6, exact

treat	guess6		Total
	0	1	
0	3	1	4
1	1	3	4
Total	4	4	8

Fisher's exact = 0.486  
 1-sided Fisher's exact = 0.243

We now summarize the general procedure of statistical hypothesis testing.

In general, **statistical hypothesis testing** consists of the following five steps:

1. Specify a **null hypothesis** and an alternative hypothesis.
2. Choose a test statistic and the **level of test  $\alpha$** .
3. Derive the **reference distribution**, which refers to the sampling distribution of the test statistic under the null hypothesis.
4. Compute the  **$p$ -value**, either one-sided or two-sided depending on the alternative hypothesis.
5. Reject the null hypothesis if the  $p$ -value is less than or equal to  $\alpha$ . Otherwise, retain the null hypothesis (i.e., fail to reject the null hypothesis).

While statistical hypothesis testing is a principled way to quantify uncertainty, the methodology has an important disadvantage. In particular, it forces researchers to make a binary decision about whether to reject the null hypothesis. In many situations, however, we are not interested in the null hypothesis itself. In fact, we may believe that the null hypothesis never strictly holds true. Instead, it could be more fruitful to quantify the degree to which the observed data deviate from the null hypothesis. In the tea-tasting experiment, we may wish to measure the extent to which the lady can taste the difference rather than simply whether or not she possesses any ability in this regard. While the  $p$ -value represents the degree to which empirical evidence refutes the null hypothesis, it does not directly correspond to the substantive quantity of interest. In other words, while hypothesis testing can determine *statistical significance*, it often fails to provide a direct measure of *scientific significance*.

### 6.2.3 ONE-SAMPLE TESTS

Using the general principle of statistical hypothesis testing we have introduced, a variety of hypothesis tests can be developed. We consider one-sample and two-sample tests, which are among the most commonly used tests. *One-sample tests* of means are used to examine the null hypothesis that the population mean equals a specific value. *Two-sample tests*, on the other hand, are based on the null hypothesis that the means of two populations equal each other. Two-sample tests are particularly useful when analyzing randomized controlled

trials, enabling researchers to investigate whether or not the observed difference in average outcomes between the treatment and control groups is likely to arise by random chance alone. These tests are used more frequently than Fisher's exact test, described earlier, because they do not rely on the sharp null hypothesis that no unit is affected by the treatment. Instead, two-sample tests concern whether treatment influences an outcome *on average*.

We start, as an example of one-sample tests, with a reanalysis of the sample surveys given in section 6.1.4. Suppose that our null hypothesis is that in the population exactly half of voters support Obama and the other half do not, i.e.,  $H_0 : p = 0.5$ . Let an alternative hypothesis be that Obama's support rate is not 0.5, i.e.,  $H_1 : p \neq 0.5$ . Now, suppose that we conduct a simple random sample and interview 1,018 selected individuals,  $n = 1,018$ . In this sample, 550 of them express support for Obama whereas the other individuals do not. This implies that the sample proportion of Obama's supporters is 54%, i.e.,  $\bar{X}_n = 550/1,018$ . Clearly, the sample proportion differs from the hypothesized proportion, 0.5, but is this difference statistically significant? Is the difference within the sampling error? Statistical hypothesis testing can answer this question.

We follow the general procedure of hypothesis testing laid out in section 6.2.2. Having defined our null and alternative hypotheses, we next choose a test statistic and the level of the test. We use the sample proportion  $\bar{X}_n$  as our test statistic and set  $\alpha = 0.05$ . We then derive the sampling distribution of this test statistic under the null hypothesis. Following the discussion in section 6.1.3 and utilizing equation (6.12), we use the central limit theorem to approximate the reference distribution of  $\bar{X}_n$  as  $\mathcal{N}(0.5, 0.5(1 - 0.5)/1,018)$ , where the variance is computed using the formula  $V(X)/n = p(1 - p)/n$ . Note that this variance of the reference distribution is constructed using Obama's support rate under the null hypothesis, i.e.,  $p = 0.5$ .

Under this setup, the *two-sided p-value*, corresponding to our null and alternative hypotheses, can be computed as the probability that under the null hypothesis we observe a value more extreme than the observed value, i.e.,  $\bar{X}_n = 550/1,018$ . Figure 6.3 shows this graphically where a more extreme value is indicated by any value either above the observed value (solid line approximately at 0.54) or below its symmetric value (dashed line approximately at 0.46). Thus, the two-sided p-value equals the sum of the blue shaded areas under the density curve. We use the `normal()` function to calculate each area. In order to compute the upper blue area in the figure, we need to subtract the result of the `normal()` function from 1.

```

. scalar xbar = 550 / 1018
. * standard deviation of sampling distribution
. scalar se = sqrt(.5 * .5 / 1018)
. * upper blue area in the figure
. scalar upper = 1 - normal((xbar - .5) / se)
. * lower blue area in the figure; identical to the upper area
. scalar lower = normal((.5 - (xbar - .5) - .5) / se)
. * two-sided p-value
. display upper + lower
.01016866

```

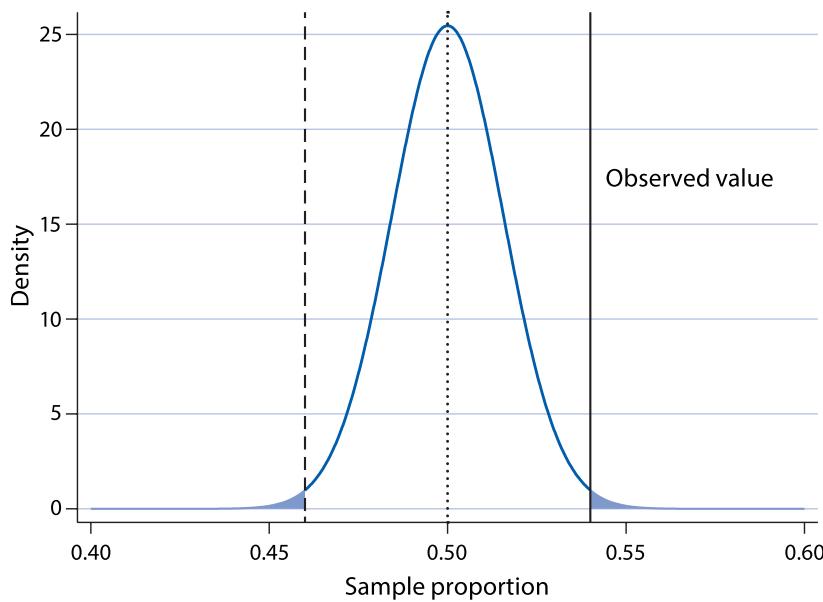


Figure 6.3. One-sided and Two-sided  $p$ -Values. The density curve represents the reference distribution under the null hypothesis that the population proportion is 0.5. The observed value is indicated by the solid vertical line. The two-sided  $p$ -value equals the sum of the two blue shaded areas under the curve whereas the one-sided  $p$ -value is equal to one of the blue areas under the curve (depending on the alternative hypothesis).

In this particular case, since both the upper and lower shaded areas have the same area (because the normal distribution is symmetric around its mean), we can simply double one of the areas to obtain the two-sided  $p$ -value. Note that this may not work in other cases where the reference distribution is not symmetric.

```
. display 2 * upper
.01016866
```

If, on the other hand, our alternative hypothesis is  $p > 0.5$  rather than  $p \neq 0.5$ , then we must compute the one-sided  $p$ -value. In this case, there is no need to consider the possibility of an extremely small value because the alternative hypothesis specifies  $p$  to be greater than the null value. Hence, the one-sided  $p$ -value is given by the blue area under the curve above the observed value in the figure.

```
. * one-sided p-value
. display upper
.00508433
```

Regardless of whether we use the one-sided or two-sided  $p$ -value, we reject the null hypothesis that Obama's support in the population is exactly 50%. We conclude that the 4 percentage point difference we observe is unlikely to arise due to chance alone.

When using the normal distribution as the reference distribution, researchers often use the  $z$ -score to standardize the test statistic by subtracting its mean and dividing it by its standard deviation. Once this transformation is made, the reference distribution becomes the standard normal distribution. That is, if we use  $\mu_0$  to denote the hypothesized mean under the null hypothesis, we have the following result so long as the sample size is sufficiently large (due to the central limit theorem):

$$\frac{\bar{X}_n - \mu_0}{\text{standard error of } \bar{X}_n} \sim \mathcal{N}(0, 1). \quad (6.19)$$

This transformation does not change the outcome of our hypothesis testing. In fact, the  $p$ -value will be identical with or without this transformation. However, one can easily compare the  $z$ -score with the critical values shown in table 6.1 in order to determine whether to reject the null hypothesis without computing the  $p$ -value. For example, under the two-sided alternative hypothesis, if the  $z$ -score is greater than 1.96, then we reject the null hypothesis. We now show, using the current example, that we obtain the same  $p$ -value as above.

```
. scalar zscore = (xbar - .5) / se
. display zscore
2.5700405

.
. * one-sided p-value
. display 1 - normal(zscore)
.00508433

.
. * two-sided p-value
. display (1 - normal(zscore)) * 2
.01016866
```

This test, which is based on the  $z$ -score of the sample mean, is called the *one-sample z-test*. Although we used this test for a Bernoulli random variable in this example, the test can be applied to a wide range of nonbinary random variables so long as the sample size is sufficiently large and the central limit theorem is applicable. For nonbinary random variables, we will use the sample variance to estimate the standard error. If the random variable  $X$  is distributed according to the normal distribution, then the same test statistic, i.e., the  $z$ -score of the sample mean, follows the  $t$ -distribution with  $n - 1$  degrees of freedom instead of the standard normal distribution. This *one-sample t-test* is more conservative than the one-sample  $z$ -test, meaning that the former gives a greater  $p$ -value than the latter. Some researchers prefer conservative inference and hence use the one-sample  $t$ -test rather than the one-sample  $z$ -test.

Suppose that  $\{X_1, X_2, \dots, X_n\}$  are  $n$  independently and identically distributed random variables with mean  $\mu$  and variance  $\sigma^2$ . The **one-sample z-test** consists of the following components:

1. Null hypothesis that the population mean  $\mu$  is equal to a prespecified value  $\mu_0: H_0: \mu = \mu_0$
2. Alternative hypothesis:  $H_1: \mu \neq \mu_0$  (two-sided),  $H_1: \mu > \mu_0$  (one-sided), or  $H_1: \mu < \mu_0$  (one-sided)
3. Test statistic ( $z$ -statistic):  $Z_n = \frac{\bar{X}_n - \mu_0}{\sqrt{\sigma^2/n}}$  where  $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$  (sample mean)
4. Reference distribution:  $Z_n \sim \mathcal{N}(0, 1)$  when  $n$  is large
5. Variance:  $\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2$  (sample variance) or  $\hat{\sigma}^2 = \mu_0(1 - \mu_0)$  if  $X$  is a Bernoulli random variable
6.  $p$ -value:  $\Phi(-|Z_n|)$  (one-sided) and  $2\Phi(-|Z_n|)$  (two-sided) where  $\Phi(\cdot)$  is the cumulative distribution function (CDF) of the standard normal distribution

If  $X$  is normally distributed, the same test statistic  $Z_n$  is called the  $t$ -statistic and follows the  $t$ -distribution with  $n - 1$  degrees of freedom. The  $p$ -value will be based on the cumulative distribution of this  $t$ -distribution. This is called the **one-sample  $t$ -test**, which is more conservative than the one-sample  $z$ -test.

There exists a general one-to-one relationship between confidence intervals and hypothesis tests. Compare equation (6.19) with equation (6.15). The difference is that the unknown population mean  $\mathbb{E}(X)$  in the former is replaced with the hypothesized population mean  $\mu_0$  in the latter. Note that under a null hypothesis the hypothesized mean  $\mu_0$  represents the actual population mean. This suggests that we reject a null hypothesis  $H_0: \mu = \mu_0$  using the  $\alpha$ -level two-sided test if and only if the  $(1 - \alpha) \times 100\%$  confidence interval does not contain  $\mu_0$ . We can confirm this result using the current example by checking that 0.5 is contained in the 99% confidence interval (we fail to reject the null hypothesis when  $\alpha = 0.01$ ) but not in the 95% confidence interval (we reject the null when  $\alpha = 0.05$ ).

```

. * 99% confidence interval contains 0.5
. display (xbar - invnormal(.995) * se), (xbar + invnormal(.995) * se)
.49990928 .58064081

.
. * 95% confidence interval does not contain 0.5
. display (xbar - invnormal(.975) * se), (xbar + invnormal(.975) * se)
.5095605 .5709896

```

It turns out that this one-to-one relationship between confidence intervals and hypothesis testing holds in general. Many researchers, however, prefer to report confidence intervals rather than  $p$ -values because the former also contain information about the magnitude of effects, quantifying *scientific significance* as well as *statistical significance*.

We conducted the one-sample  $z$ -test for sample proportion “by hand” in order to illustrate the underlying idea. However, Stata has the `prtest` command, which enables us to conduct this test in a single line of code. For a one-sample test of sample proportion, we can use the immediate version of the command (`prtest i`, which is followed respectively by the number of trials, the proportion of successes, and the success probability of the null hypothesis).

The default confidence level is 95%, which we can change with the `level()` option. We could also enter in the number of successes rather than the proportion and specify the `count` option to let Stata know we are using counts.

```
. * one-sample proportions test with 95% confidence interval
. prtesti 1018 550 .5, count

One-sample test of proportion                                x: Number of obs = 1018

```

	Mean	Std. Err.	[95% Conf. Interval]
x	.540275	.0156201	.5096603 .5708898

p = proportion(x) z = 2.5700  
Ho: p = 0.5  
Ha: p < 0.5 Ha: p != 0.5 Ha: p > 0.5  
Pr(Z < z) = 0.9949 Pr(|Z| > |z|) = 0.0102 Pr(Z > z) = 0.0051

The `prtest` command could also be used to test the equality of proportions across two variables or for a single variable in two groups. In the former case, the syntax of the command is `prtest var1 == var2`. We could also test `var1` against a specific value of the null hypothesis by including the proportion rather than `var2`. The syntax to test whether a proportion is the same across groups is `prtest var1, by(groupvar)` where `groupvar` is a dichotomous variable.

As we can see, the `prtest` command also conveniently yields confidence intervals. Note that the standard error used for confidence intervals is different from the standard error used for hypothesis testing. This is because the latter standard error is derived under the null hypothesis  $\sqrt{p(1-p)/n}$  whereas the standard error for confidence intervals is computed using the estimated proportion,  $\sqrt{\bar{X}_n(1-\bar{X}_n)/n}$ . To illustrate a different level of confidence intervals, we can compute 99% confidence intervals as follows:

```
. * one-sample proportions test with 99% confidence interval
. prtesti 1018 550 .5, count level(99)

One-sample test of proportion                                x: Number of obs = 1018

```

	Mean	Std. Err.	[99% Conf. Interval]
x	.540275	.0156201	.5000404 .5805096

p = proportion(x) z = 2.5700  
Ho: p = 0.5  
Ha: p < 0.5 Ha: p != 0.5 Ha: p > 0.5  
Pr(Z < z) = 0.9949 Pr(|Z| > |z|) = 0.0102 Pr(Z > z) = 0.0051

As another example, we revisit the analysis of the STAR project given in section 6.1.5. We first conduct a one-sample  $t$ -test just for illustration. Suppose that we test the null hypothesis that the population mean test score is 710, i.e.,  $H_0 : \mu = 710$ . We use the `ttest` command where we specify the null value  $\mu_0$  using an equals sign ( $=$ ). The other arguments such as `level()` and `by()` work in the exact same way as for the `prtest` command. We use the reading test score for our analysis and conduct a two-sided one-sample  $t$ -test. As the result below shows, we retain, at the 0.05 level, the null hypothesis that the population mean test score is 710.

```
. * two-sided one-sample t-test
. use STAR, clear
. ttest g4reading = 710

One-sample t test



| Variable | Obs   | Mean     | Std. Err. | Std. Dev. | [95% Conf. Interval] |
|----------|-------|----------|-----------|-----------|----------------------|
| g4read_g | 2,353 | 721.2478 | 1.080775  | 52.42592  | 719.1284 723.3671    |


mean = mean(g4reading) t = 10.4071
Ho: mean = 710 degrees of freedom = 2352
Ha: mean < 710 Ha: mean != 710 Ha: mean > 710
Pr(T < t) = 1.0000 Pr(|T| > |t|) = 0.0000 Pr(T > t) = 0.0000
```

#### 6.2.4 TWO-SAMPLE TESTS

We now move to a more realistic analysis of the STAR project. When analyzing randomized controlled trials like this, researchers often conduct a statistical hypothesis test with the null hypothesis that the population average treatment effect (PATE) is zero, i.e.,  $H_0 : \mathbb{E}(Y_i(1) - Y_i(0)) = 0$  with a two-sided alternative hypothesis given by  $H_1 : \mathbb{E}(Y_i(1) - Y_i(0)) \neq 0$ . If we assume that the PATE cannot be negative, then we employ a one-sided alternative hypothesis,  $H_1 : \mathbb{E}(Y_i(1) - Y_i(0)) > 0$ . In contrast, if we assume that the PATE cannot be positive, we set  $H_1 : \mathbb{E}(Y_i(1) - Y_i(0)) < 0$ . In this application, we would like to test whether or not the PATE of small class size on the grade-four reading score (relative to regular class size) is zero.

To test this null hypothesis, we use the difference-in-means estimator as a test statistic. More generally, beyond randomized controlled trials, we can use the two-sample tests based on the difference-in-means estimator to investigate the null hypothesis that the means are equal between these two populations. What is the reference distribution of this test statistic? We can approximate it by appealing to the central limit theorem as in section 6.1.5. The theorem implies that the sample means of the treatment and control groups have a normal distribution. Therefore, under the null hypothesis of equal means between the two populations, the difference between these two sample means is also normally distributed with mean zero. Furthermore, the  $z$ -score of the difference in sample means follows the standard normal distribution. We can use this fact to conduct the *two-sample z-test* (see equation (6.18) for the

expression of standard error, which serves as the denominator of the test statistic). As in the one-sample tests, if the outcomes are assumed to be normally distributed, the *two-sample t-test* can be used, which yields a more conservative inference.

Suppose that  $\{X_1, X_2, \dots, X_{n_0}\}$  represent  $n_0$  independently and identically distributed random variables with mean  $\mu_0$  and variance  $\sigma_0^2$ . Similarly,  $\{Y_1, Y_2, \dots, Y_{n_1}\}$  represent  $n_1$  independently and identically distributed random variables with mean  $\mu_1$  and variance  $\sigma_1^2$ . The **two-sample z-test** of sample means consists of the following components:

1. Null hypothesis that two populations have the same mean:  $H_0: \mu_0 = \mu_1$
2. Alternative hypothesis:  $H_1: \mu_0 \neq \mu_1$  (two-sided),  $H_1: \mu_0 > \mu_1$  (one-sided), or  $H_1: \mu_0 < \mu_1$  (one-sided)
3. Test statistic (*z*-statistic):  $Z_n = (\bar{Y}_{n_1} - \bar{X}_{n_0}) / \sqrt{\frac{1}{n_1} \hat{\sigma}_1^2 + \frac{1}{n_0} \hat{\sigma}_0^2}$
4. Reference distribution:  $Z_n \sim \mathcal{N}(0, 1)$  when  $n_0$  and  $n_1$  are large
5. Variance:  $\hat{\sigma}_0^2 = \frac{1}{n_0-1} \sum_{i=1}^{n_0} (X_i - \bar{X}_{n_0})^2$  and  $\hat{\sigma}_1^2 = \frac{1}{n_1-1} \sum_{i=1}^{n_1} (Y_i - \bar{Y}_{n_1})^2$  (sample variances) or  $\hat{\sigma}_0^2 = \hat{\sigma}_1^2 = \hat{p}(1 - \hat{p})$  with  $\hat{p} = \frac{n_0}{n_0+n_1} \bar{X}_{n_0} + \frac{n_1}{n_0+n_1} \bar{Y}_{n_1}$  if  $X$  and  $Y$  are Bernoulli random variables
6. *p*-value:  $\Phi(-|Z_n|)$  (one-sided) and  $2\Phi(-|Z_n|)$  (two-sided), where  $\Phi(\cdot)$  is the cumulative distribution function (CDF) of the standard normal distribution

If  $X$  and  $Y$  are normally distributed, the same test statistic  $Z_n$  is called the *t*-statistic and follows the *t*-distribution. The *p*-value will be based on the cumulative distribution of this *t*-distribution. This is called the **two-sample t-test**, which is more conservative than the one-sample *z*-test..

Recall from section 6.1.5 that the estimated PATE is stored as scalar `ate_est`, and its standard error is given by the scalar `ate_se`. Using these objects, we compute the one-sided and two-sided *p*-values as follows:

```
. * one-sided p-value
. display normal(-abs(ate_est) / ate_se)
.09350361

.
. * two-sided p-value
. display 2 * normal(-abs(ate_est) / ate_se)
.18700723
```

Since this *p*-value is much greater than the typical threshold of 5%, we cannot reject the hypothesis that the average treatment effect of small class size on the fourth-grade reading test score is zero.

Our hypothesis test is based on the large sample approximation because we relied upon the central limit theorem to derive the reference distribution. Similar to the discussion in section 6.1.5, if we assume that the outcome variable is normally distributed, then we could use the *t*-distribution instead of the normal distribution to conduct a hypothesis test. As a test statistic, we use the *z*-score for the difference-in-means estimator, which is called the *t*-statistic in the case of this two-sample *t*-test. Unlike the one-sample example discussed in section 6.1.5, however, the degrees of freedom must be approximated for the *two-sample t-test*. Because the *t*-distribution generally has heavier tails than the normal distribution, the *t-test* is more conservative and hence is often preferred even when the outcome variable may not be normally distributed.

In Stata, we can conduct a two-sample *t*-test by the `ttest` command as we did for a one-sample *t*-test. For the two-sample *t*-test, the command can take either two variables to test for differences across those variables, or it can take one variable and assess differences across two groups. As we saw earlier in this chapter, the latter case uses the `by()` option to specify the two groups. In addition, the `ttest` command assumes that the two groups in the *t*-test have equal variances. We can relax that assumption by specifying the `unequal` option (for the Satterthwaite approximation of the degrees of freedom) or the `welch` option (for Welch's formula for the degrees of freedom).

```
. * testing the null of zero average treatment effect
. ttest g4reading if classtype == 1 | classtype == 2, by(classtype) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
1	726	723.3912	1.913012	51.54494	719.6355 727.1469
2	836	719.89	1.83885	53.16788	716.2806 723.4993
combined	1,562	721.5173	1.326654	52.4322	718.9151 724.1195
diff		3.501232	2.653485		-1.703591 8.706055

```
diff = mean(1) - mean(2)                                     t = 1.3195
Ho: diff = 0                                              Satterthwaite's degrees of freedom = 1541.25
Ha: diff < 0                                              Ha: diff != 0
Pr(T < t) = 0.9064                                         Pr(|T| > |t|) = 0.1872
                                                               Pr(T > t) = 0.0936
```

The output displays the value of the *t*-statistic as well as the *p*-value and the degrees of freedom for Student's *t*-distribution used for the test. Since the *p*-value is greater than the standard threshold of  $\alpha = 0.05$ , we fail to reject the null hypothesis that the average treatment effect of small class size on the fourth grade reading score is zero. As in the case of `prtest`, the output of the `ttest` command contains the confidence interval for the corresponding level. We focus on values in the `diff` row. As expected from the use of the *t*-distribution, this confidence interval is slightly wider than the confidence interval based on the normal

approximation we obtained in section 6.1.5. The confidence interval also contains zero, which is consistent with the fact that we fail to reject the null hypothesis of zero average treatment effect.

As another application of hypothesis tests, we reanalyze the labor market discrimination experiment described in section 2.1. In this experiment, fictitious résumés of job applicants were sent to potential employers. Researchers randomly assigned stereotypically African American or Caucasian names to each résumé and examined whether or not the callback rate depended on the race of the applicant. The data set we analyze is contained in the data file `resume.dta`. The names and descriptions of variables in this data set are given in table 2.1. The outcome variable of interest is `call`, which indicates whether or not each résumé received a callback. The treatment variable is the race of the applicant, `race`, and we focus on the comparison between black-sounding and white-sounding names.

We test the null hypothesis that the probability of receiving a callback is the same between résumés with black-sounding names and those with white-sounding names. We use the `prtest` command to implement the two-sample *z*-test. The first input is the `call` variable. We then specify the `race` variable within the `by()` option so that we get the proportion of callbacks by race. Stata shows the *p*-value under the null hypothesis of equal means (two-sided test) as well as the *p*-value under the null hypothesis of one mean being greater than the other (one-sided test). We will use a one-sided test because résumés with black-sounding names are hypothesized to receive fewer callbacks.

```
. use resume, clear
. prtest call, by(race)

Two-sample test of proportions                               black: Number of obs = 2435
                                                               white: Number of obs = 2435



| Group | Mean                   | Std. Err. | z     | P> z     | [95% Conf. Interval] |
|-------|------------------------|-----------|-------|----------|----------------------|
| black | .0644764               | .0049771  |       | .0547214 | .0742314             |
| white | .0965092               | .0059841  |       | .0847807 | .1082378             |
| diff  | -.0320329              | .0077834  |       | -.047288 | -.0167777            |
|       | under H <sub>0</sub> : | .0077969  | -4.11 | 0.000    |                      |



diff = prop(black) - prop(white)                                z = -4.1084  

H0: diff = 0



|              |                    |               |                        |              |                    |
|--------------|--------------------|---------------|------------------------|--------------|--------------------|
| Ha: diff < 0 | Pr(Z < z) = 0.0000 | Ha: diff != 0 | Pr( Z  >  z ) = 0.0000 | Ha: diff > 0 | Pr(Z > z) = 1.0000 |
|--------------|--------------------|---------------|------------------------|--------------|--------------------|


```

The result supports the alternative hypothesis that résumés with white-sounding names are more likely to receive callbacks than those with black-sounding names. It is instructive to directly compute this *p*-value without using the `prtest` command. Under the null hypothesis of equal proportions between the two groups, i.e.,  $H_0: \mu_0 = \mu_1$ , the standard error

of the difference-in-means (or more accurately difference-in-proportions) estimator can be computed as

$$\sqrt{\frac{\widehat{V}(X)}{n_0} + \frac{\widehat{V}(Y)}{n_1}} = \sqrt{\frac{\widehat{p}(1 - \widehat{p})}{n_0} + \frac{\widehat{p}(1 - \widehat{p})}{n_1}} = \sqrt{\widehat{p}(1 - \widehat{p}) \left( \frac{1}{n_0} + \frac{1}{n_1} \right)}, \quad (6.20)$$

where  $X$  and  $Y$  are the outcome variables for the résumés with black-sounding and white-sounding names, respectively,  $n_0$  and  $n_1$  are sample sizes, and  $\widehat{p} = \frac{1}{n_0+n_1} (\sum_{i=1}^{n_0} X_i + \sum_{i=1}^{n_1} Y_i)$  is the overall sample proportion. We use the same estimate  $\widehat{p}(1 - \widehat{p})$  for the variances of  $X$  and  $Y$  because under the null hypothesis of identical proportions, their variances, which are based on the proportions, are also identical.

```
. summarize call

Variable |   Obs    Mean   Std. Dev.   Min   Max
-----+-----+-----+-----+-----+-----+
call     | 4,870  .0804928  .2720826  0      1

. scalar p = r(mean)

.

. * sample size and proportions
. summarize call if race == "black"

Variable |   Obs    Mean   Std. Dev.   Min   Max
-----+-----+-----+-----+-----+-----+
call     | 2,435  .0644764  .2456501  0      1

. scalar n0 = r(N)

. scalar p0 = r(mean)

. summarize call if race == "white"

Variable |   Obs    Mean   Std. Dev.   Min   Max
-----+-----+-----+-----+-----+-----+
call     | 2,435  .0965092  .295349  0      1

. scalar n1 = r(N)

. scalar p1 = r(mean)

.

. * point estimate
. scalar est = p1 - p0

. display est
.03203285

.

. * standard error
. scalar std_err = sqrt(p * (1 - p) * (1 / n0 + 1 / n1))
```

```

. display std_err
.00779689

.
. * z-statistic
. scalar zstat = est / std_err

. display zstat
4.1084122

.
. * one-sided p-value
. display normal(-abs(zstat))
.00001992

```

The exact same  $z$ -statistic and  $p$ -value are obtained as when we used the `prtest` command.

### 6.2.5 PITFALLS OF HYPOTHESIS TESTING

Since Fisher's tea-tasting experiment, hypothesis testing has been extensively used in the scientific community to determine whether or not empirical findings are statistically significant. Statistical hypothesis testing represents a rigorous methodology to draw a conclusion in the presence of uncertainty. However, the prevalent use of hypothesis testing also leads to *publication bias* because only statistically significant results, and especially the ones that are surprising to the scientific community, tend to be published. In many social science journals, the  $\alpha$ -level of 5% is regarded as the cutoff that determines whether empirical findings are statistically significant or not. As a result, researchers tend to submit their papers to journals only when their empirical results have  $p$ -values smaller than this 5% threshold. In addition, journals may be more likely to publish statistically significant results than nonsignificant results. This is problematic because even if the null hypothesis is true, researchers have a 5% chance of obtaining a  $p$ -value less than 5%.

In one study, two researchers examined more than 100 articles published in the two leading political science journals over a decade or so.<sup>2</sup> The researchers collected the  $p$ -values for the hypotheses tested in those articles. Figure 6.4 shows that a majority of reported findings have  $p$ -values less than or equal to the 5% threshold, which is indicated by the blue vertical line. There also appears to be a discontinuous jump at the threshold, suggesting that journals are publishing more empirical results that are just below the threshold than results just above it.

Another important pitfall regarding hypothesis testing is *multiple testing*. Recall that statistical hypothesis testing is probabilistic. We never know with 100% certainty whether the null hypothesis is true. Instead, as explained earlier, we typically have type I and type II errors when conducting hypothesis tests (see table 6.3). Multiple testing problems refer to the possibility of *false discoveries* when testing multiple hypotheses.

To see this, suppose that a researcher tests 10 hypotheses when, unbeknown to the researcher, all of these hypotheses are in fact false. What is the probability that the researcher

<sup>2</sup> Alan Gerber and Neil Malhotra (2008). "Do statistical reporting standards affect what is published? Publication bias in two leading political science journals." *Quarterly Journal of Political Science*, vol. 3, no. 3, pp. 313–326.

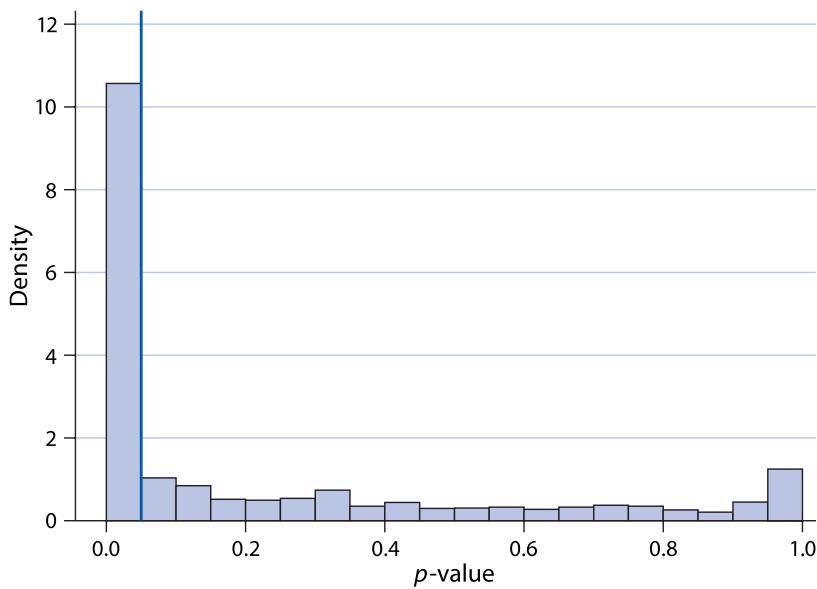


Figure 6.4. The Distribution of  $p$ -values for Hypothesis Tests Published in Two Leading Political Science Journals.

rejects at least one null hypothesis using 5% as the threshold? If we assume independence among these hypotheses tests, we can compute this probability as

$$\begin{aligned} P(\text{reject at least one hypothesis}) &= 1 - P(\text{reject no hypothesis}) \\ &= 1 - 0.95^{10} \approx 0.40. \end{aligned}$$

The second equality follows because the probability of not rejecting the null hypothesis when the null hypothesis is true is  $1 - \alpha = 0.95$  and we assume independence among these 10 hypothesis tests. The researcher thus has a 40% chance of making at least one false discovery. The lesson here is that if we conduct many hypothesis tests, we are likely to falsely find statistically significant results.

To illustrate the multiple testing problem, consider “Paul the Octopus” shown in figure 7.5a. This octopus in a German aquarium attracted media attention during the 2010 World Cup soccer tournament by correctly predicting all seven matches involving Germany, as well as the outcome of the final match between the Netherlands and Spain. Paul predicted by choosing to enter one of two containers with a country flag as shown in the figure. Given these data, we can conduct a hypothesis test with the null hypothesis that Paul does not possess any ability to predict soccer matches. Under this null hypothesis, Paul randomly guesses a winner out of two countries in question. What is the probability that Paul correctly predicts the outcomes of all eight matches? Since Paul has a 50% chance of correctly predicting each match, this one-sided  $p$ -value is equal to  $1/2^8 \approx 0.004$ . This value is well below the usual 5% threshold and hence can be considered statistically significant.

However, the problem of multiple testing suggests that if we have many animals predict soccer matches, we are likely to find an animal that appears to be prophetic. During the same

(a) Paul the octopus

(a) Mani the parakeet

Figure 6.5. Two Animal Oracles that Correctly Predicted the Outcomes of Soccer Matches.  
Sources: (a) Reuters/Wolfgang Rattay. (b) AP Images/Joan Leong.

World Cup, another animal, “Mani the Parakeet” shown in figure 7.5b, was reported to have a similar oracle ability. The parakeet correctly predicted only six out of eight matches. Each time, he selected one of two pieces of paper with his beak and flipped it to reveal a winner, without viewing country flags as Paul did. Since no scientific theory suggests animals can possess such predictive ability, we may conclude that Paul and Mani represent false discoveries due to the problem of multiple testing. Although beyond the scope of this book, statisticians have developed various methods that make appropriate adjustments for multiple testing.

The **multiple testing problem** is that conducting many hypothesis tests is likely to result in false discoveries, i.e., incorrect rejection of null hypotheses.

#### 6.2.6 POWER ANALYSIS

Another problem of hypothesis testing is that null hypotheses are often not interesting. For example, who would believe that the small class in the STAR study has *exactly* zero average causal effect on students’ test scores as assumed under the null hypothesis? The effect size might be small, but it is hard to imagine that it is exactly zero. A related problem is that failure to reject the null hypothesis does not necessarily mean that the null hypothesis is true. Failure to reject the null may arise because data are not informative about the null hypothesis. For example, if the sample size is too small, then even if the true average treatment effect is not zero, researchers may fail to reject the null hypothesis of zero average effect because the standard error is too large.

We use *power analysis* in order to formalize the degree of informativeness of data in hypothesis tests. The *power* of a statistical hypothesis test is defined as 1 minus the probability of *type II error*:

$$\text{power} = 1 - P(\text{type II error}).$$

Recall from the discussion in section 6.2.2 that type II error occurs when researchers retain a false null hypothesis. Therefore, we would like to maximize the power of a statistical

hypothesis test so that we can detect departure from the null hypothesis as much as possible.

Power analysis is often used to determine the smallest sample size necessary to estimate the parameter with enough precision that its observed value is distinguishable from the parameter value assumed under the null hypothesis. This is typically done as part of research planning in order to inform data collection. In sample surveys, for example, researchers wish to know the number of people they must interview in order to reject the null hypothesis of an exact tie in support level when one candidate is ahead of the other by a prespecified degree (see also the discussion in section 6.1.4). Moreover, experimentalists use power analysis to compute the number of observations necessary to reject the null hypothesis of zero average treatment effect when the effect is actually not zero. As a result, power analysis is often required for research grant applications in order to justify the budget that researchers are requesting.

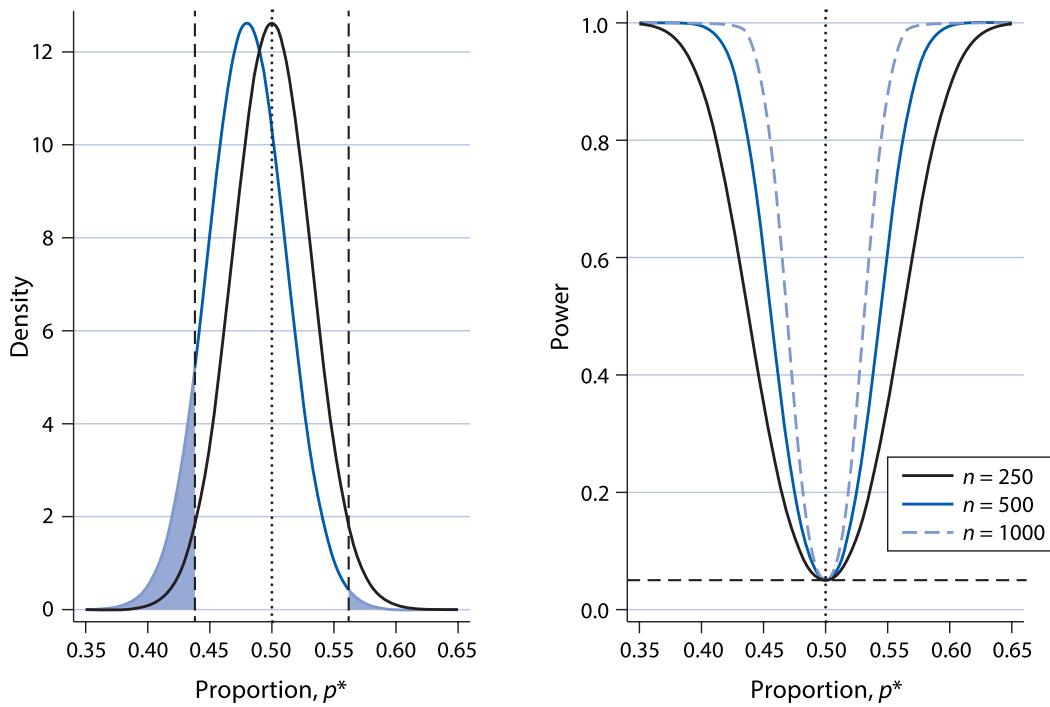
Again, we use survey sampling as an example. Suppose that we wish to find out how many respondents we must interview to be able to reject the null hypothesis that the support level for Obama, denoted by  $p$ , is exactly 50% when the true support level is at least 2 percentage points away from an exact tie, i.e., 48% or less, or 52% or greater. That is, 2 percentage points is the smallest deviation from the null hypothesis we would like to detect with a high probability. Further assume that we will use the sample proportion as the test statistic, and that the significance level is set to  $\alpha = 0.05$  with a two-sided alternative hypothesis.

To compute the power, we need to consider two sampling distributions of the test statistic. The first is the sampling distribution under the null distribution. We have already derived the large sample approximation of this sampling distribution earlier:  $\mathcal{N}(p, p(1 - p)/n)$ , where  $p$  is the null value of the population proportion. In our application,  $p = 0.5$ . The second is the sampling distribution under a hypothetical data-generating process. In the current case, this distribution is approximated by  $\mathcal{N}(p^*, p^*(1 - p^*)/n)$  via the *central limit theorem*, where  $p^*$  is either less than or equal to 0.48 or greater than or equal to 0.52.

The left-hand plot of figure 6.6 graphically illustrates the mechanics of power analysis in this case. In the plot, the two sampling distributions of the sample proportion, one centered around 0.5 under the null hypothesis (black solid line) and the other centered around 0.48 under a hypothetical data-generating process (blue solid line), are shown. We choose 0.48 as the mean value under the hypothetical data-generating process because any distribution with a mean less than this value would result in greater statistical power, which is the probability of correctly rejecting the null, and hence would require a smaller sample size. For the meantime, we set the sample size  $n$  to 250.

Under this setting, we compute the power of the statistical test, which is the probability of rejecting the null hypothesis. To do this, we first derive the thresholds that determine the rejection region. As shown in section 6.2.3, the threshold is equal to the null value  $p_0$  plus or minus the product of the standard error and critical value  $z_{\alpha/2}$ , i.e.,  $p_0 \pm z_{\alpha/2} \times$  standard error, where in the current setting  $p_0 = 0.5$  and  $z_{\alpha/2} \approx 1.96$ . In the left-hand plot of the figure, these thresholds are denoted by black dashed lines and we reject the null hypothesis if an observed value is more extreme than they are.

We use the probability distribution indicated by the blue solid line in the figure when computing the probability of rejection under the hypothetical data-generating process. That is, the power of the test equals the sum of the two blue shaded areas in the figure, one large area below the lower threshold and the other small area above the upper threshold. Formally,



**Figure 6.6. Illustration of Power Analysis.** In the left-hand plot, the solid black line represents the sampling distribution of sample proportion under the null hypothesis  $p=0.5$  (vertical dotted line). The blue solid line represents the sampling distribution of the test statistic under a hypothetical data-generating process, which has mean 0.48. The sum of the two blue shaded areas equals the power of this statistical test when the significance level is  $\alpha=0.05$ . The vertical dashed lines represent thresholds, above or below which the null hypothesis will be rejected. The right-hand plot displays the power function under the same setting with three different sample sizes.

it is given by

$$\text{power} = P(\bar{X}_n < p - z_{\alpha/2} \times \text{standard error}) + P(\bar{X}_n > p + z_{\alpha/2} \times \text{standard error}).$$

In this equation, the sample proportion  $\bar{X}_n$  is assumed to be approximately distributed according to  $\mathcal{N}(p^*, p^*(1-p^*)/n)$ , where in the current application  $p^*$  is set to 0.48. We can compute the power of a test in Stata as follows:

```

. * set the parameters
. scalar n = 250
. scalar pstar = .48 // data-generating process
. scalar pnull = .5 // null value
.
. * standard errors under the hypothetical data-generating process
. scalar sestar = sqrt(pstar * (1 - pstar) / n)

```

```

. * standard error under the null
. scalar se = sqrt(pnull * (1 - pnull) / n)
. * power
. display normal(((pnull - invnormal(.975) * se) - pstar) / sestar) + ///
>           (1 - normal(((pnull + invnormal(.975) * se) - pstar) / sestar))
.09673114

```

Under these conditions, the power of the test is only 10%. We can examine how the power of this test changes as a function of the sample size and hypothetical data-generating process. The right-hand plot of figure 6.6 presents the *power function*, where the horizontal axis represents the population proportion under the hypothetical data-generating process and each line indicates a different sample size. We observe that the power of a statistical test increases as the sample size becomes greater and the true population proportion  $p^*$  shifts away from the null value  $p = 0.5$ .

This specific example illustrates the main principle of power analysis. We summarize the general procedure below.

**Power** is defined as the probability of rejecting the null hypothesis when the null hypothesis is false, which is equal to 1 minus the probability of type II error. **Power analysis** consists of the following steps:

1. Select the settings of the statistical hypothesis test you plan to use. This includes the specification of the test statistic, null and alternative hypotheses, and significance level.
2. Choose the population parameter value under a hypothetical data-generating process.
3. Compute the probability of rejecting the null hypothesis under this data-generating process with a given sample size.

One can then vary the sample size to examine how the power of the test changes to decide the **sample size** necessary for the desired level of power.

The power analysis can be conducted in a similar manner for two-sample tests. Consider the two-sample test of proportions, which can be used to analyze a randomized experiment with a binary outcome variable. The test statistic is the difference in sample proportion between the treatment and control groups,  $\bar{Y}_{n_1} - \bar{X}_{n_0}$ . Under the null hypothesis that this difference in the population, or the population average treatment effect (PATE), is equal to zero, the sampling distribution of the test statistic is given by  $\mathcal{N}(0, p(1-p)(1/n_1 + 1/n_0))$ , where  $p$  is the overall population proportion (see equation (6.20)), which is equal to the weighted average of the proportions in the two groups,  $p = (n_0 p_0 + n_1 p_1) / (n_0 + n_1)$ . To compute the power of the statistical test in this case, we must specify the population proportion separately for the treatment and control groups,  $p_1^*$  and  $p_0^*$ , under a hypothetical data-generating process. Then, the sampling distribution of the test statistic under this data-generating process is given

by  $\mathcal{N}(p_1^* - p_0^*, p_1^*(1 - p_1^*)/n_1 + p_0^*(1 - p_0^*)/n_0)$ . Using this information, we can compute the probability of rejecting the null.

As an example, consider the résumé experiment analyzed in section 2.1. Suppose that we plan to send out 500 résumés with black-sounding names and another 500 résumés with white-sounding names. Further, assume that we expect the callback rate to be around 5% for black names and 10% for white names.

```
. * specify the parameters
. scalar n1 = 500
. scalar n0 = 500
. scalar p1star = .05
. scalar p0star = .1
```

To calculate the power of this statistical test, we first compute the overall callback rate as a weighted average of callback rates of the two groups, where the weights are their sample size. We then compute the standard error under the null hypothesis, i.e., standard error =  $\sqrt{p(1 - p)(1/n_0 + 1/n_1)}$ , as well as under the hypothetical data-generating process, i.e., standard error\* =  $\sqrt{p_1^*(1 - p_1^*)/n_1 + p_0^*(1 - p_0^*)/n_0}$ .

```
. * overall call back rate as a weighted average
. scalar p = (n1 * p1star + n0 * p0star) / (n1 + n0)
. * standard error under the null
. scalar std_err = sqrt(p * (1 - p) * (1 / n1 + 1 / n0))
. * standard error under the hypothetical data-generating process
. scalar sestar = sqrt(p1star * (1 - p1star) / n1 + p0star * (1 - p0star) / n0)
```

We can now compute the power by calculating the probability that the difference in two proportions,  $\bar{Y}_n - \bar{X}_n$ , takes a value either less than  $-z_{\alpha/2} \times \text{standard error}$  or greater than  $-z_{\alpha/2} \times \text{standard error}^*$ , under the hypothetical data-generating process.

```
. display normal((-invnormal(.975) * std_err - (p1star - p0star)) / sestar) + ///
> 1 - normal((invnormal(.975) * std_err - (p1star - p0star)) / sestar)
.85227997
```

While for illustration we computed the power by hand, we can use the `power` command available in Stata. This command, which is applicable to the two-sample test for proportions and for *Student's t-test*, can either compute the power given a set of parameters or determine a parameter value given a target power level. Depending on the type of power, the arguments of this command may include the sample size per group (`n()`), population proportions for two groups (`p1.star` and `p2.star`), significance level (`alpha()`), and power (`power()`). Note that the command assumes two groups have an identical sample size, i.e.,  $n_0 = n_1$ .

Stata has many different types of power available through the `power` command. Because we are interested in a two-sample proportion test here, we use the `twoprop` option after `power`. The following syntax gives a result identical to our previous calculation.

```
. power twoprop 0.05 .1, n(1000)

Estimated power for a two-sample proportions test
Pearson's chi-squared test
Ho: p2 = p1 versus Ha: p2 != p1

Study parameters:

    alpha =      0.0500
        N =      1,000
    N per group =      500
        delta =      0.0500  (difference)
          p1 =      0.0500
          p2 =      0.1000

Estimated power:

    power =      0.8523
```

The `power` command also enables sample size calculation by simply setting the `power()` argument to a desired level and omitting the `n()` option. For example, if we want to know, under the same conditions as above, the minimum sample size necessary to obtain the 90% level of power, we use the syntax that follows. The result implies that we need at least 582 observations per group in order to achieve the 90% level of power.

```
. power twoprop 0.05 .1, p(.9)

Performing iteration ...

Estimated sample sizes for a two-sample proportions test
Pearson's chi-squared test
Ho: p2 = p1 versus Ha: p2 != p1

Study parameters:

    alpha =      0.0500
    power =      0.9000
        delta =      0.0500  (difference)
          p1 =      0.0500
          p2 =      0.1000

Estimated sample sizes:

        N =      1,164
    N per group =      582
```

For continuous variables, we can conduct power analysis based on *Student's t-test* introduced in Section 6.2.4. The logic is exactly the same as that described above for one-sample and two-sample tests of proportions. The `power` command can perform a power analysis where the `method` argument specifies a two-sample (`twomeans`) or one-sample (`onemean`) test. For a one-sample *t*-test, we must specify the hypothesized mean, the true or target mean, and the standard deviation (`sd`) of a normal random variable under a hypothetical data-generating process. For a two-sample *t*-test, the command assumes that the standard deviation and sample size are identical for the two groups. We, therefore, specify the true means for each group under a hypothetical data-generating process as well as a standard deviation.

We present two examples of using the `power` command based on the *t*-test. The first is the power calculation for a one-sample test with a true mean of 0.25 and standard deviation of 1. The sample size is 100. Recall that the assumed mean value under the null hypothesis is zero so we enter that as the first value in the command.

```
. power onemean 0 .25, n(100) sd(1)

Estimated power for a one-sample mean test
t test
Ho: m = m0  versus  Ha: m != m0

Study parameters:
    alpha =      0.0500
        N =        100
    delta =      0.2500
        m0 =      0.0000
        ma =      0.2500
        sd =      1.0000

Estimated power:
    power =      0.6970
```

Under this setting, the power is calculated to be 70%. What is the sample size we need to have a power of 0.9 under the same setting? We can answer this question by specifying the `power()` argument in the `power onemean` command while leaving the `n` argument unspecified.

```
. power onemean 0 .25, power(.9) sd(1)

Performing iteration ...

Estimated sample size for a one-sample mean test
t test
Ho: m = m0  versus  Ha: m != m0

Study parameters:
    alpha =      0.0500
    power =      0.9000
```

```

delta = 0.2500
m0 = 0.0000
ma = 0.2500
sd = 1.0000

Estimated sample size:

N = 171

```

The minimum sample size for obtaining a power of 0.9 or greater is 171. The second example is the sample size calculation for a one-sided two-sample test with a true mean difference of 0.25 and standard deviation of 1. We set the desired power to be 90%. We also add `onesided` as an option to derive the minimum sample size if the mean of `m2` is greater than the mean of `m1`.

```

. power twomeans 0 .25, power(.9) onesided
Performing iteration ...

Estimated sample sizes for a two-sample means test
t test assuming sd1 = sd2 = sd
Ho: m2 = m1 versus Ha: m2 > m1

Study parameters:

alpha = 0.0500
power = 0.9000
delta = 0.2500
m1 = 0.0000
m2 = 0.2500
sd = 1.0000

Estimated sample sizes:

N = 550
N per group = 275

```

The result shows that we need a minimum of 275 observations per group to achieve the power of 90% in this setting.

### 6.3 Linear Regression Model with Uncertainty

As the final topic of this chapter, we consider the uncertainty of estimates based on the linear regression model introduced in chapter 4. In that chapter, we used the linear regression model mainly as a tool to make predictions. We also showed that when applied to a randomized controlled trial with binary treatments, the linear regression model can yield unbiased estimates of average treatment effects. In this section, we introduce another perspective that portrays the linear regression model as an approximation of the *data-generating process* in the real world. Under this framework, we can quantify the uncertainty of our estimates over

repeated hypothetical sampling from the specified generative model. Once we view the linear regression model as a generative model, we can compute the standard errors and confidence intervals for our quantities of interest and conduct hypothesis testing.

### 6.3.1 LINEAR REGRESSION AS A GENERATIVE MODEL

Recall that the linear regression model with  $p$  predictors (explanatory or independent variables) is defined as

$$Y_i = \alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \epsilon_i. \quad (6.21)$$

In this model,  $Y$  represents the outcome or response variable,  $X_{ij}$  is the  $j$ th predictor, for  $j = 1, 2, \dots, p$ , and  $\epsilon_i$  denotes the unobserved error term for the  $i$ th observation. The model also has a total of  $(p + 1)$  coefficients to be estimated, where  $\alpha$  represents an intercept and  $\beta_j$  denotes a coefficient for the  $j$ th explanatory variable for  $j = 1, 2, \dots, p$ .

According to this model, the outcome variable is generated as a linear function of the explanatory variables and the error term. For example, in section 4.2, we modeled the relationship between facial impressions and election outcomes using linear regression. In that application, the election outcome was a linear function of facial impressions and the error term. The error term contains all determinants of election outcomes that we do not observe, such as campaign resources, name recognition, and voter mobilization efforts.

In the model, the only variable we do not directly observe is the error term. As such, the key assumption of the model concerns the distribution of this random variable  $\epsilon_i$ . Specifically, the linear regression model is based on the following *exogeneity* assumption.

The **exogeneity** assumption for the **linear regression model** is defined as

$$\mathbb{E}(\epsilon_i | \mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_p) = \mathbb{E}(\epsilon_i) = 0. \quad (6.22)$$

The assumption implies that the unobserved determinants of outcome, contained in the error term  $\epsilon_i$ , are unrelated to all the observed predictors  $X_{ij}$  for  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, p$ . In this equation,  $\mathbf{X}_j$  is an  $n \times 1$  vector containing the  $j$ th covariate of all observations.

The assumption says that the *conditional expectation* of the error term given the explanatory variables, which is the first term in equation (6.22), is equal to its marginal or unconditional expectation, which is the second term in the equation and is equal to zero. The marginal expectation of the error term can always be assumed to be zero in the linear regression model so long as an intercept  $\alpha$  is included in the model. The exogeneity assumption implies that the mean of the error term does not depend on the predictors or explanatory variables included in the model. In other words, the unobserved determinants of the outcome variable, which are contained in the error term, should be *uncorrelated* with all of the observed predictors. In the election example, this implies that other, unobserved determinants of election outcomes should not be correlated with candidates' facial impressions.

In general, the conditional expectation of a random variable  $Y$  given another random variable  $X$ , denoted by  $\mathbb{E}(Y | X)$ , is the expectation of  $Y$  given a particular value of  $X$ . As such, this conditional expectation is a function of  $X$ , i.e.,  $\mathbb{E}(Y | X) = g(X)$ , where  $g(X)$  is called the *conditional expectation function*. All the definitions and rules of expectation introduced in section 5.3.5 hold for conditional expectation, except that we treat the variables in the conditioning set as fixed and compute the expectation with respect to the conditional distribution of  $Y$  given  $X$ . Thus, under the exogeneity assumption, the linear regression model assumes that the conditional expectation function for the outcome variable given the set of predictors is linear:

$$\mathbb{E}(Y_i | \mathbf{X}_1, \dots, \mathbf{X}_p) = \alpha + \beta_1 X_{i1} + \dots + \beta_p X_{ip}.$$

When deriving this result, we used the exogeneity assumption as well as the fact that the conditional expectation of  $\beta_j X_{ij}$  given  $\mathbf{X}_1, \dots, \mathbf{X}_p$  equals itself.

The **conditional expectation** of a random variable  $Y$  given another random variable  $X$  is denoted by  $\mathbb{E}(Y | X)$  and is defined as

$$\mathbb{E}(Y | X) = \begin{cases} \sum_y y \times f(y | X) & \text{if } Y \text{ is discrete,} \\ \int y \times f(y | X) dy & \text{if } Y \text{ is continuous,} \end{cases}$$

where  $f(Y | X)$  is the conditional probability mass function (conditional probability density function) of the discrete (continuous) random variable  $Y$  given  $X$ .

In *randomized controlled trials*, a violation of exogeneity does not occur because treatment assignment is randomized. In the framework of the linear regression model, this means that the treatment variable, which is represented by  $X$ , is statistically independent of all observed and unobserved pretreatment characteristics, which are contained in  $\epsilon$ . Therefore, the exogeneity assumption is automatically satisfied. Consider the randomized controlled trial about women as policymakers described in section 4.3.1. In this experiment, the explanatory variable of interest,  $X$ , is whether seats in the local government, Gram Panchayat (GP), are reserved for female leaders. This variable is randomized and hence statistically independent of all other possible determinants of policy outcomes. For example, the number of new or repaired drinking water facilities in the village is likely to be determined not only by the existence of female leaders but also by numerous other factors such as the population size and the income level. Fortunately, we do not have to worry about these potential *unobserved confounders* because the randomized treatment assignment makes the treatment variable independent of these factors.

In *observational studies*, however, the exogeneity assumption may be violated. Suppose that the reservation of some GPs for female leaders is not randomized. Then, it is possible that villages with high levels of education and liberal ideologies are likely to elect female leaders for their GPs. Under this scenario, we cannot simply attribute the difference in the number

of new or repaired drinking water facilities between villages to the gender of their politicians alone. It may be that highly educated villagers want better drinking water facilities and politicians are simply responding to the demands of their constituency. That is, both female and male politicians are responding to their constituencies, but their policy outcomes are different because they have different constituencies rather than because their genders are different. In observational studies, the unobserved confounders may be contained in the error term (e.g., education level of villagers), and if they are correlated with the observed explanatory variables (e.g., gender of politicians), the exogeneity assumption will be violated.

How can we address this problem of unobserved confounding in observational studies? In chapter 2, we learned that one strategy is to compare the treated units with similar control units. Ideally, we would like to find units that did not receive treatment and yet are similar to the treated units in terms of many observed characteristics. In the study on the minimum wage and employment described in section 2.5, researchers chose fast-food restaurants in Pennsylvania (PA), in which the minimum wage was not increased, as the control group for the fast-food restaurants in New Jersey (NJ), for which the minimum wage was raised. The idea was that since these restaurants are quite similar in their patterns of employment, products, and sales, we can use the restaurants in PA to infer the employment level of the restaurants in NJ that would have resulted if the minimum wage had not been increased. If there exist no unobserved factors, other than the treatment in NJ, that influence employment in NJ fast-food restaurants (i.e., no unobserved confounders), then the average difference in employment between the restaurants in NJ and those in PA can be attributed to the increase in NJ's minimum wage. The assumption of no unobserved confounding factors has several different names, including *unconfoundedness*, *selection on observables*, and *no omitted variables*, but they all mean the same thing.

The assumption of no unobserved confounding factors studied in chapter 2, therefore, is directly related to the exogeneity assumption under the linear regression model. Indeed, the exogeneity assumption will be violated whenever unobserved confounding variables exist. In the linear regression model framework, we can address this problem by measuring these confounders and including them as additional predictors in the model in order to adjust for their differences between the treatment and control groups. Although this strategy assumes a linear relationship between the outcome and these confounding variables, conceptually it is the same as comparing treated and control units that have similar characteristics. It can be shown that so long as all confounding variables are included in the model (and the linear relationship between the outcome and all explanatory variables holds), the estimated coefficient for the treatment variable represents an unbiased estimate of the average treatment effect.

In the minimum-wage example, assume that the only confounding factors between the fast-food restaurants in NJ and those in PA are the fast-food chain to which each restaurant belongs, its wage, and the proportion of full-time employment before the minimum wage was increased in NJ. We adjust for these three variables in the linear model, where the outcome variable is the proportion of full-time employment after the minimum wage was increased in NJ and the treatment variable is whether a restaurant is located in NJ. We use the data set described in table 2.5 and regress the outcome variable and three confounding variables using the `regress` command. Before we fit the linear regression, we compute the proportion of full-time employment before and after the minimum wage was increased in NJ. We also

create an indicator, or “dummy” variable, that equals 1 if a restaurant is located in NJ and 0 if it is in PA.

```
. use minwage, clear
. * proportion of full-time employment before minimum-wage increase
. generate fullpropbefore = fullbefore / (fullbefore + partbefore)
. * same thing after minimum-wage increase
. generate fullpropaftter = fullaftter / (fullaftter + partaftter)
. * an indicator for NJ: 1 if it's located in NJ and 0 if in PA
. generate nj = cond(location != "PA", 1, 0)
```

We now regress the proportion of full employment after the minimum-wage increase on the treatment variable (i.e., whether a restaurant is located in NJ) as well as three other potential confounding variables. We note that `chain` is a string variable with four different chains of fast-food restaurants. We must first convert it into a numeric variable using the `encode` command before it can be included in the regression. When a factor variable is used in the `regress` command with the `i.` prefix, as we saw in section 4.3.2, the command will automatically create the appropriate number of indicator variables for each category. In this case, since we have an intercept and the factor has four categories, the function will create three indicator variables. The `regress` command by default includes an intercept. If we remove the intercept using the `nocons` option, then it will create one indicator variable for each of the four categories. We also have to tell Stata to not include a base category. We do this with the `i.bn.` option. As explained in section 4.3.2, these two models are equivalent and yield an identical predicted value given the same values of the explanatory variables while yielding different estimates of coefficients. We use the `estimates store name` command to store the estimation results in a matrix that we will access again in section 6.3.4. Storing results is useful for the efficient comparison of estimates across models and data sets, for creating tables, and for truncating code by alleviating the need to recode or generate already coded variables when reopening a file and rerunning a model. This capability also allows one to perform the same calculations on multiple model results using loops, for example. We can view the stored regression coefficients by typing `matrix list e(b)`. The variance-covariance matrix of the estimators is found in matrix `e(V)`.

```
. encode chain, generate(chainnum)
. regress fullpropaftter nj fullpropbefore wagebefore ibn.chainnum, nocons
```

Source	SS	df	MS	Number of obs	=	358
Model	36.2837318	7	5.18339026	F(7, 351)	=	87.21
Residual	20.8616238	351	.059434826	Prob > F	=	0.0000
Total	57.1453556	358	.159623898	R-squared	=	0.6349
				Adj R-squared	=	0.6277
				Root MSE	=	.24379

fullpropaftter	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
nj	.05422	.0332086	1.63	0.103	-.0110929 .119533
fullpropbefore	.168794	.0566188	2.98	0.003	.0574391 .2801489
wagebefore	.0813345	.0389222	2.09	0.037	.0047843 .1578846
chainnum					
burgerking	-.1156253	.1788751	-0.65	0.518	-.467427 .2361765
kfc	-.1508	.1831042	-0.82	0.411	-.5109193 .2093194
roys	-.2063862	.1867076	-1.11	0.270	-.5735924 .1608201
wendys	-.2201326	.188401	-1.17	0.243	-.5906695 .1504043

```

. estimates store minwage_noint
. matrix list e(b)
e(b) [1,7]
1.          2.          3.
      nj  fullpropbe~e  wagebefore  chainnum  chainnum  chainnum
y1   .05422001    .168794    .08133446   -.11562526  -.15079995  -.20638616
4.
      chainnum
y1   -.2201326

```

The result shows that the minimum wage increase in NJ raised the proportion of full employees by 5.4 percentage points (represented by the estimated coefficient for the NJ variable) after adjusting for the proportion of full-time employees and wages before the minimum wage increase as well as the chains of fast-food restaurants. By excluding the intercept, we can immediately compare the estimated coefficients across fast-food restaurant chains. We find that Burger King is predicted to have the highest proportion of full-time employment after adjusting for the other factors in the model. If we include an intercept, the estimated coefficients need to be interpreted relative to the base category, which will be dropped from the regression model. The base category of a factor variable represents a category to which the other categories of the variable are compared.

. regress fullpropaftter nj fullpropbefore wagebefore i.chainnum						
Source	SS	df	MS	Number of obs	=	358
Model	1.56949138	6	.261581897	F(6, 351)	=	4.40
Residual	20.8616238	351	.059434826	Prob > F	=	0.0003
Total	22.4311152	357	.062832256	R-squared	=	0.0700
				Adj R-squared	=	0.0541
				Root MSE	=	.24379

fullpropaftter	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
nj	.05422	.0332086	1.63	0.103	-.0110929 .119533
fullpropbefore	.168794	.0566188	2.98	0.003	.0574391 .2801489
wagebefore	.0813345	.0389222	2.09	0.037	.0047843 .1578846
chainnum					
kfc	-.0351747	.0348688	-1.01	0.314	-.1037528 .0334034
roys	-.0907609	.0336256	-2.70	0.007	-.1568939 -.0246279
wendys	-.1045073	.0418232	-2.50	0.013	-.186763 -.0222517
_cons	-.1156253	.1788751	-0.65	0.518	-.467427 .2361765

```
. estimates store minwage_int
```

The `regress` command excluded the indicator variable for Burger King (the base category) from the regression, which means that the estimated coefficients for all other fast-food restaurant chains are relative to Burger King. Consistent with the previous result, we find that all other estimated coefficients are negative, indicating that Burger King is predicted to have the highest proportion of full-time employment after adjusting for the other factors in the model. We emphasize that these two models are equivalent, yielding the same predicted values. To demonstrate, we use the outputs of the two regression models to predict the outcome for the first observation in the data. The predicted values are identical.

```
. quietly regress fullpropaftter nj fullpropbefore wagebefore ibn.chainnum, nocons
. predict xb_noint, xb
. display xb_noint
.27093673

.
. quietly regress fullpropaftter nj fullpropbefore wagebefore i.chainnum
. predict xb_int, xb
. display xb_int
.27093673
```

Valid inference under the linear model assumes the **exogeneity assumption** given in equation (6.22). This assumption will be violated if there exist **unobserved confounders**. To make the exogeneity assumption more plausible, researchers can measure confounding variables and include them as additional explanatory variables in the linear regression model.

### 6.3.2 UNBIASEDNESS OF ESTIMATED COEFFICIENTS

How accurately can we estimate the coefficients of the linear regression model? Under the assumption that the linear regression model actually describes the true data-generating process, we consider the question of how to quantify the uncertainty associated with estimated coefficients. For simplicity, let us consider the model with one predictor only, though the results presented in this section can be generalized to linear regression with more than one predictor:

$$Y_i = \alpha + \beta X_i + \epsilon_i. \quad (6.23)$$

Recall from the discussion given in section 4.2.3 that if the linear regression model contains only an intercept and one predictor, then the least squares estimates are given by

$$\hat{\alpha} = \bar{Y} - \hat{\beta} \bar{X}, \quad (6.24)$$

$$\hat{\beta} = \frac{\sum_{i=1}^n (Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}. \quad (6.25)$$

In this equation,  $\bar{X}$  and  $\bar{Y}$  represent the sample average of the predictor  $X_i$  and the outcome variable  $Y_i$ , respectively.

It turns out that under the exogeneity assumption these least squares coefficients,  $\hat{\alpha}$  and  $\hat{\beta}$ , are unbiased for their corresponding true values,  $\alpha$  and  $\beta$ , respectively. Formally, we may write  $\mathbb{E}(\hat{\alpha}) = \alpha$  and  $\mathbb{E}(\hat{\beta}) = \beta$ . This means that if we generate the data according to this linear model, the least squares estimates of the coefficients will equal their true values, on average, across the hypothetically repeated data sets. Consequently, the method of least squares produces unbiased estimates while minimizing the sum of squared residuals.

For those who are mathematically inclined, we show this important result analytically. Since we assume that the linear regression model is the true data-generating process, we substitute the linear model expression given in equation (6.23) into equation (6.24). Noting that the average outcome is given by  $\bar{Y} = \alpha + \beta \bar{X} + \bar{\epsilon}$ , we obtain the following expression for the estimated intercept:

$$\hat{\alpha} = \alpha + \beta \bar{X} + \bar{\epsilon} - \hat{\beta} \bar{X} = \alpha + (\beta - \hat{\beta}) \bar{X} + \bar{\epsilon}.$$

This equation shows that the estimation error  $\hat{\alpha} - \alpha$  is given by  $(\beta - \hat{\beta}) \bar{X} + \bar{\epsilon}$ . Similarly, we use equation (6.23) to rewrite the estimated slope coefficient given in equation (6.25) as the sum of the true value  $\beta$  and the estimation error  $\hat{\beta} - \beta$ :

$$\hat{\beta} = \frac{\sum_{i=1}^n (\beta X_i + \epsilon_i - \beta \bar{X} - \bar{\epsilon})(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} = \beta + \underbrace{\frac{\sum_{i=1}^n (\epsilon_i - \bar{\epsilon})(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}}_{\text{estimation error}},$$

where we used the fact that  $\sum_{i=1}^n \beta X_i = \sum_{i=1}^n \beta \bar{X}$ .

We can further simplify the numerator of this estimation error, i.e., the second term in this equation:

$$\begin{aligned}\sum_{i=1}^n (\epsilon_i - \bar{\epsilon})(X_i - \bar{X}) &= \sum_{i=1}^n \epsilon_i(X_i - \bar{X}) - \sum_{i=1}^n \bar{\epsilon}(X_i - \bar{X}) \\ &= \sum_{i=1}^n \epsilon_i(X_i - \bar{X}) - \bar{\epsilon} \underbrace{\left( \sum_{i=1}^n X_i - n\bar{X} \right)}_{=0} \\ &= \sum_{i=1}^n \epsilon_i(X_i - \bar{X}).\end{aligned}$$

We then obtain the following final expression for the estimation error of the slope coefficient:

$$\hat{\beta} - \beta = \frac{\sum_{i=1}^n \epsilon_i(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}. \quad (6.26)$$

As discussed in section 6.1.1, to prove the unbiasedness of  $\hat{\beta}$ , we must show that on average  $\hat{\beta}$  equals its true value  $\beta$  over repeated (hypothetical) data-generating processes. Mathematically, we compute the expectation of  $\hat{\beta}$  and show it is equal to  $\beta$ , i.e.,  $\mathbb{E}(\hat{\beta}) = \beta$ . In this case, we first compute the conditional expectation of  $\hat{\beta}$  given the explanatory variable vector  $X$  under the exogeneity assumption given in equation (6.22), then show  $\mathbb{E}(\hat{\beta} | X) = \beta$ . This means that for a given value of  $X$ , we consider the hypothetical process of repeatedly generating the outcome variable  $Y$  by sampling the error term  $\epsilon$  independent of  $X$  and then compute the least squares estimates  $\hat{\alpha}$  and  $\hat{\beta}$ . While these estimates differ each time, on average they should equal the true values  $\alpha$  and  $\beta$ , respectively.

We first calculate the conditional expectation of the estimated slope coefficient. Since the expectation is computed given the predictor vector  $X$ , the only random variable is the error term  $\epsilon$ . This means that the other terms can be considered as constants and taken out of the expectation:

$$\mathbb{E}(\hat{\beta} - \beta | X) = \frac{1}{\sum_{i=1}^n (X_i - \bar{X})^2} \sum_{i=1}^n \mathbb{E}(\epsilon_i | X)(X_i - \bar{X}) = 0.$$

The second equality is implied by the exogeneity assumption  $\mathbb{E}(\epsilon | X) = 0$ . Therefore, the estimated slope coefficient for  $X_i$  is unbiased conditional on the predictor. Using this result, we can also show that the estimated intercept is unbiased conditional on the predictor vector  $X$ :

$$\mathbb{E}(\hat{\alpha} - \alpha | X) = \mathbb{E}(\hat{\beta} - \beta | X)\bar{X} + \mathbb{E}(\bar{\epsilon} | X) = 0.$$

The result follows from the fact that  $\mathbb{E}(\hat{\beta} - \beta | X) = 0$  (unbiasedness of  $\hat{\beta}$ ) and  $\mathbb{E}(\bar{\epsilon} | X) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}(\epsilon_i | X) = 0$  (exogeneity). Since this means that given *any* value of the predictor vector  $X$  the estimated coefficients,  $\hat{\alpha}$  and  $\hat{\beta}$ , are unbiased, conditional unbiasedness implies unbiasedness without conditioning, i.e.,  $\mathbb{E}(\hat{\alpha}) = \alpha$  and  $\mathbb{E}(\hat{\beta}) = \beta$ .

Under the exogeneity assumption, the least squares estimates of the coefficients in the linear regression model are unbiased.

The argument we just made, that conditional unbiasedness of estimated coefficients implies (unconditional) unbiasedness, can be made more generally and is called the *law of iterated expectation*.

The **law of iterated expectation** states that for any two random variables  $X$  and  $Y$ , the following equality holds:

$$\mathbb{E}(Y) = \mathbb{E}\{\mathbb{E}(Y | X)\}.$$

The inner expectation averages over  $Y$  given  $X$ , yielding a function of  $X$ , and the outer expectation averages this resulting conditional expectation function over  $X$ .

As an example, let  $Y$  be an individual income and  $X$  be the racial group to which the individual belongs. To obtain the average income in a population, we could simply compute the mean of everyone's income  $\mathbb{E}(Y)$  or first compute the average income for each racial category  $\mathbb{E}(Y | X) = g(X)$  and then obtain the overall mean income by calculating the weighted average of race-specific means, where the weight is proportional to the size of racial group  $\mathbb{E}(g(X))$ . Applying the law of iterated expectation, we would formally conclude that the estimated coefficients are unbiased:

$$\mathbb{E}(\hat{\alpha}) = \mathbb{E}\{\mathbb{E}(\hat{\alpha} | X)\} = \mathbb{E}(\alpha) = \alpha,$$

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\{\mathbb{E}(\hat{\beta} | X)\} = \mathbb{E}(\beta) = \beta.$$

### 6.3.3 STANDARD ERRORS OF ESTIMATED COEFFICIENTS

Now that we have established the unbiasedness of estimated coefficients, we consider their standard errors. The standard error of each estimated coefficient represents the (estimated) standard deviation of its sampling distribution (see section 6.1.2). The sampling distribution is produced through a hypothetically repeated sampling process, yielding different estimated coefficients across samples. The standard error quantifies the average variability of the estimated coefficient over this repeated sampling procedure.

As in the case of unbiasedness, we consider the linear regression model with one predictor for the sake of simplicity. We derive the variance of the sampling distribution of the estimated slope coefficient  $\hat{\beta}$  and then take its square root to obtain the standard error. As in the case of bias, we first compute the conditional variance given the predictor  $X$ . Recall the discussion in section 5.3.5 that the variance of a random variable does not change even if we add a constant to it. Thus, the variance of the estimated coefficient  $\hat{\beta}$  equals that of the estimation error  $\hat{\beta} - \beta$  since  $\beta$  is an (albeit unknown) constant. Using equation (6.26), we obtain

$$\begin{aligned}
\mathbb{V}(\hat{\beta} | \mathbf{X}) &= \mathbb{V}(\hat{\beta} - \beta | \mathbf{X}) \\
&= \mathbb{V}\left(\frac{\sum_{i=1}^n \epsilon_i(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} \mid \mathbf{X}\right) \\
&= \frac{1}{\{\sum_{i=1}^n (X_i - \bar{X})^2\}^2} \mathbb{V}\left(\sum_{i=1}^n \epsilon_i(X_i - \bar{X}) \mid \mathbf{X}\right). \tag{6.27}
\end{aligned}$$

The third equality follows from equation (5.38) and the fact that the denominator is a function only of the predictor  $X$ , which is treated as a constant when computing the conditional variance given  $\mathbf{X}$ .

To further simplify the expression in equation (6.27), we assume *homoskedasticity* of the error term. That is, we assume that, conditional on the predictor  $X$ , the error term of observation  $i$  is independent of that of another observation, and that the variance of the error term does not depend on the predictor  $X$ .

The assumption of **homoskedastic error** consists of the following two components:

1.  $\epsilon_i$  is independent of  $\epsilon_j$  conditional on  $\mathbf{X}$  for all  $i \neq j$ .
2. The variance of error does not depend on the predictor:  $\mathbb{V}(\epsilon_i | \mathbf{X}) = \mathbb{V}(\epsilon_i)$ .

Under this homoskedasticity assumption, we can further simplify the numerator of equation (6.27):

$$\mathbb{V}\left(\sum_{i=1}^n \epsilon_i(X_i - \bar{X}) \mid \mathbf{X}\right) = \sum_{i=1}^n \mathbb{V}(\epsilon_i | \mathbf{X})(X_i - \bar{X})^2 = \mathbb{V}(\epsilon_i) \sum_{i=1}^n (X_i - \bar{X})^2. \tag{6.28}$$

Putting this together with equation (6.27), we arrive at the following variance of the estimated slope coefficient  $\hat{\beta}$  under the homoskedasticity and exogeneity assumptions:

$$\mathbb{V}(\hat{\beta} | \mathbf{X}) = \frac{\mathbb{V}(\epsilon_i)}{\sum_{i=1}^n (X_i - \bar{X})^2}. \tag{6.29}$$

Although the above expression represents the conditional variance of  $\hat{\beta}$  given the predictor  $X$ , we can also compute the unconditional variance of  $\hat{\beta}$ . The former is based on the variability of  $\hat{\beta}$  under the hypothetical scenario of repeated sampling of  $Y_i$  given  $X_i$  (or equivalently  $\epsilon_i$  given  $X_i$  because  $Y_i$  is a function of  $X_i$  and  $\epsilon_i$ ) for each observation, where  $X_i$  is fixed throughout. In contrast, the latter represents the uncertainty of  $\hat{\beta}$  under a somewhat more natural data-generating process where  $Y_i$  and  $X_i$  (or equivalently  $\epsilon_i$  and  $X_i$ ) are jointly sampled from the population for each hypothetical realization of the data. To derive the unconditional variance of  $\hat{\beta}$ , we use the following *law of total variance*.

The **law of total variance** states that for any two random variables  $X$  and  $Y$  the following equality holds:

$$\mathbb{V}(Y) = \mathbb{V}\{\mathbb{E}(Y | X)\} + \mathbb{E}\{\mathbb{V}(Y | X)\}.$$

The first term represents the variance of conditional expectation and the second term represents the expectation of conditional variance.

This law implies that the unconditional variance of random variable  $Y$  is equal to the sum of the variance of the conditional expectation of  $Y$  given  $X$  and the expectation of the conditional variance of  $Y$  given  $X$ . Applying the law of total variance, we can show that the unconditional variance of  $\hat{\beta}$  can be derived as

$$\begin{aligned} \mathbb{V}(\hat{\beta}) &= \mathbb{V}(\mathbb{E}(\hat{\beta} | \mathbf{X})) + \mathbb{E}\{\mathbb{V}(\hat{\beta} | \mathbf{X})\} \\ &= \underbrace{\mathbb{V}(\beta)}_{=0} + \mathbb{E} \left( \frac{\mathbb{V}(\epsilon_i)}{\sum_{i=1}^n (x_i - \bar{X})^2} \right) \\ &= \mathbb{V}(\epsilon_i) \mathbb{E} \left[ \frac{1}{\sum_{i=1}^n (X_i - \bar{X})^2} \right]. \end{aligned} \quad (6.30)$$

In the above equation,  $\mathbb{V}(\beta) = 0$  because  $\beta$  is a constant. This implies that the unconditional variance of  $\hat{\beta}$  is equal to the expected value of the conditional variance of  $\hat{\beta}$ , i.e.,  $\mathbb{V}(\hat{\beta}) = \mathbb{E}\{\mathbb{V}(\hat{\beta} | \mathbf{X})\}$ . In turn, a good estimate of the conditional variance is also a good estimate of the unconditional variance.

Given this result, under the assumption of homoskedastic error, we can compute the standard error of  $\hat{\beta}$  as an estimate of the unconditional variance given in equation (6.30). We do this by first estimating  $\mathbb{V}(\epsilon_i)$  using the sample variance of residuals  $\hat{\epsilon}_i = Y_i - \hat{\alpha} - \hat{\beta}X_i$ , and then taking the square root of it. That is, if we denote the estimated conditional variance by  $\widehat{\mathbb{V}}(\hat{\beta})$ , then the standard error of  $\hat{\beta}$  is

$$\text{standard error of } \hat{\beta} = \sqrt{\widehat{\mathbb{V}}(\hat{\beta})} = \sqrt{\frac{\frac{1}{n} \sum_{i=1}^n \hat{\epsilon}_i^2}{\sum_{i=1}^n (X_i - \bar{X})^2}}. \quad (6.31)$$

When estimating  $\mathbb{V}(\epsilon_i)$ , we used the fact that the sample mean of residuals is always zero,<sup>3</sup> i.e.,  $\frac{1}{n} \sum_{i=1}^n (\hat{\epsilon}_i - \bar{\epsilon})^2 = \frac{1}{n} \sum_{i=1}^n \hat{\epsilon}_i^2$ .

These standard errors are based on the assumption of homoskedastic errors. If this assumption is violated, then the calculation of standard errors needs to be adjusted. For example, in randomized controlled trials, the variance may differ for the treatment and control

<sup>3</sup>Since the expectation of the error term is also zero and hence does not need to be estimated, we divide the sum of squared residuals by  $n$  instead of  $n - 1$  often used for the sample variance calculation (see the discussion in section 2.6.2).

groups. In fact, when we computed the standard error for the difference-in-means estimator, we separately calculated the variance for each group (see equation (6.18)). If the error variance depends on the predictor, we say that error is *heteroskedastic*. Although beyond the scope of this book, there are various ways to compute the standard errors that account for heteroskedastic errors. They are called *heteroskedasticity-robust standard errors*.

### 6.3.4 INFERENCE ABOUT COEFFICIENTS

Given the computed standard error, we can calculate the confidence intervals following the procedure described in section 6.1.3. Specifically, using the *central limit theorem*, we can show that as the sample size increases, the sampling distribution of  $\hat{\beta}$  approaches a normal distribution centered around the mean:

$$\text{z-score of } \hat{\beta} = \frac{\hat{\beta} - \beta}{\text{standard error of } \hat{\beta}} \underset{\text{approx.}}{\sim} \mathcal{N}(0, 1). \quad (6.32)$$

We can thus use the critical values based on the standard normal distribution to construct the  $(1 - \alpha) \times 100\%$  level confidence interval below:

$$\text{CI}(\alpha) = [\hat{\beta} - z_{\alpha/2} \times \text{standard error}, \hat{\beta} + z_{\alpha/2} \times \text{standard error}]. \quad (6.33)$$

We can also conduct a hypothesis test for the slope coefficient. In some cases, we may want to test the null hypothesis that the slope coefficient is equal to a particular value  $\beta_0$ . Most often, researchers use zero as the true value under the null hypothesis and ask whether or not the true coefficient for the predictor is equal to zero, i.e.,  $\beta_0 = 0$ . Under the general hypothesis-testing framework developed in section 6.2, our null hypothesis is  $H_0: \beta = \beta_0$ . The test statistic is the z-score, i.e.,  $z^* = (\hat{\beta} - \beta_0) / \text{standard error}$ , and the sampling distribution of this test statistic  $z^*$  under the null hypothesis is the standard normal distribution. Hence, we can compute the  $p$ -value using the CDF of the standard normal distribution. Accordingly, the two-sided  $p$ -value is given by  $2 \times P(Z \leq z^*)$ , where  $Z$  is a standard normal random variable.

Just as for the analysis of randomized experiments, researchers often use a more conservative confidence interval based on Student's  $t$ -distribution (see section 6.1.5). Technically, if we make an additional assumption that the error term is normally distributed with mean zero and homoskedastic variance, then the sampling distribution of  $z^*$ , which is called the  *$t$ -statistic* in this setting, is given by, without approximation, Student's  $t$ -distribution with  $n - 2$  degrees of freedom. This contrasts with the asymptotic approximation based on the standard normal distribution without assuming a particular distribution for the error term. The degrees of freedom are  $n - 2$  because two parameters,  $\alpha$  and  $\beta$ , are estimated from the data. Since Student's  $t$ -distribution has thicker tails than the standard normal distribution, we will have a greater critical value and as a result, obtain a wider confidence interval and a greater  $p$ -value.

We illustrate these calculations with two examples. First, we revisit the randomized experiment from chapter 4, examining the effects of women as policymakers in India (see section 4.3.1). We again use the data set `women.dta`. The variable names and descriptions are given in table 4.7. Recall that we used these data to regress the number of drinking

water facilities in a village (`water`) on a binary variable (`reserved`) that indicated whether a GP reserved a council position for women.

. use women, clear						
. regress water reserved						
Source	SS	df	MS	Number of obs	=	322
Model	6144.58583	1	6144.58583	F(1, 320)	=	5.49
Residual	357956.337	320	1118.61355	Prob > F	=	0.0197
Total	364100.922	321	1134.27079	R-squared	=	0.0169
				Adj R-squared	=	0.0138
				Root MSE	=	33.446
water	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reserved	9.252423	3.947746	2.34	0.020	1.485608	17.01924
_cons	14.73832	2.2863	6.45	0.000	10.24024	19.2364

Rerunning our regression model, we find that the point estimate of the slope coefficient is 9.252 and its standard error is 3.948. This output uses a conservative confidence interval based on Student's *t*-distribution. The *t*-statistic for the estimated slope coefficient is, therefore, 2.34. If the null hypothesis is that the slope coefficient is zero, then the two-sided *p*-value can be computed using Student's *t*-distribution with 320 degrees of freedom because the sample size is 322. In the summary output, this *p*-value is shown to be .02. Using the  $\alpha = 0.05$  level of statistical significance, we then reject the null hypothesis that the slope coefficient is zero. The output table also displays the confidence intervals, with the default set at 95%. The level of statistical significance can be changed by specifying the `level()` option in the `regress` command where a numeric value goes in the parentheses.

The result suggests that having the GP reserved for women is estimated to increase the number of water drinking facilities by 9.25 facilities with a 95% confidence interval of [1.49, 17.02]. As expected, we observe that the 95% confidence interval does not contain zero.

While beyond the scope of this book, we can also compute the standard error and confidence interval of the estimated coefficients in a more general setting with multiple predictors. We return to the minimum wage example from section 6.3.1, where we used `estimates store` to save the estimation results of our model. The `estimates replay` command allows us to see the results again and the `estimates restore` command reactivates them. Once active, we can perform many of the same calculations as we would after running the actual regression. We can access the coefficients using notation `_b` with the variable of interest placed in brackets, or access the standard error with `_se`. Here, we restore the minimum wage regression results from the model that excludes the intercept. When using the `estimates` commands, it is no longer necessary to first open the raw data file (in this case, `minwage.dta`). To see a list of the model estimates currently stored in memory, type `estimates dir`.

```

. estimates restore minwage_noint
(results minwage_noint are active now)

. * retrieve coefficient and standard error for nj variable
. display _b[nj]
.05422001

. display _se[nj]
.03320865

```

In this observational study, we are interested in the average effect, which corresponds to the coefficient of the `nj` variable. The average effect of the minimum-wage increase on the proportion of full-time employees in NJ is estimated to be 5.4 percentage points with a standard error of 3.3 percentage points. According to the result, we fail to reject the null hypothesis that the average effect of the minimum-wage increase is zero. In other words, we cannot preclude the possibility that the nonzero point estimate we obtained may be due to the sampling error under the scenario that the minimum-wage increase did not, on average, change the proportion of full-time employment. The  $p$ -values in this case are based on Student's  $t$ -distribution with 351 degrees of freedom because we have a total of 358 observations and 7 parameters to be estimated.

As shown on page 341, the confidence interval contains zero, consistent with the result of the hypothesis test. However, a large portion of the confidence interval contains positive values, providing evidence that the minimum-wage increase in NJ may not have decreased the proportion of full-time employment.

We can also see the *root-mean-squared-error* (RMSE), or the sample standard deviation of residuals. Since there are  $(p + 1)$  parameters to be estimated, the degrees of freedom equal  $(n - p - 1)$  instead of the usual  $(n - 1)$  used when computing the average. The root mean square error represents the average magnitude of residuals under the fitted model. The output also includes the  $R^2$ , or *coefficient of determination*, which represents the proportion of explained variation in the outcome (see section 4.2.5). As explained in section 4.3.2, the *adjusted R<sup>2</sup>* includes the adjustment due to the degrees of freedom, penalizing models with large numbers of predictors.

### 6.3.5 INFERENCE ABOUT PREDICTIONS

As shown in chapter 4, one of the main advantages of regression modeling is its ability to predict outcomes of interest. In the case of linear regression models, once we estimate the coefficients, we can use the model to predict an outcome variable given the values of the predictors in the model. Below, we show how to compute the standard error and construct confidence intervals for a prediction based on the linear regression model.

For the sake of simplicity, consider the linear regression model with a single predictor, i.e.,  $Y_i = \alpha + \beta X_i + \epsilon_i$ . We are interested in obtaining the standard error of the predicted value from this model when the predictor  $X$  takes a particular value  $x$ :

$$\hat{Y} = \hat{\alpha} + \hat{\beta}x.$$

To derive the variance of the predicted value  $\hat{Y}$ , we must recognize the fact that  $\hat{\alpha}$  and  $\hat{\beta}$  are possibly correlated with each other. When two random variables,  $X$  and  $Y$ , are correlated, the variance of their sum is not equal to the sum of their variances. Instead, the variance of their sum includes their *covariance*, defined as follows.

Let  $X$  and  $Y$  be random variables. Their **covariance** is defined as

$$\begin{aligned}\text{Cov}(X, Y) &= \mathbb{E}\{(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))\} \\ &= \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y).\end{aligned}$$

The **correlation**, a standardized version of covariance, is given by

$$\text{Cor}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\mathbb{V}(X)\mathbb{V}(Y)}}.$$

*Sample correlation*, or the correlation of a sample, was introduced in chapter 3 (see Section 3.6.2). If the two random variables are independent of each other, their covariance and correlation are zero. In addition, the general formula for the variance of the sum of two (possibly dependent) random variables is given as

$$\mathbb{V}(X + Y) = \mathbb{V}(X) + \mathbb{V}(Y) + 2 \text{Cov}(X, Y).$$

More generally,

$$\mathbb{V}(aX + bY + c) = a^2\mathbb{V}(X) + b^2\mathbb{V}(Y) + 2ab \text{Cov}(X, Y),$$

where  $a, b, c$  are constants.

Since  $\hat{\alpha}$  and  $\hat{\beta}$  may not be independent, using the general formula, we obtain the following variance of predicted value  $\hat{Y}$  when the predictor  $X$  equals a particular value  $x$ :

$$\mathbb{V}(\hat{Y}) = \mathbb{V}(\hat{\alpha} + \hat{\beta}x) = \mathbb{V}(\hat{\alpha}) + \mathbb{V}(\hat{\beta})x^2 + 2x \text{Cov}(\hat{\alpha}, \hat{\beta}).$$

We can compute the standard error by estimating each component of this variance and then taking the square root of the estimated variance of  $\hat{Y}$ :

$$\text{standard error of } \hat{Y} = \sqrt{\widehat{\mathbb{V}(\hat{\alpha})} + \widehat{\mathbb{V}(\hat{\beta})}x^2 + 2x \widehat{\text{Cov}(\hat{\alpha}, \hat{\beta})}}.$$

Once the standard error is calculated, we can apply the *central limit theorem* to approximate the sampling distribution of the  $z$ -score for the predicted value  $\hat{Y}$  using the standard normal distribution:

$$\text{z-score of } \widehat{Y} = \frac{\widehat{Y} - (\alpha + \beta x)}{\text{standard error of } \widehat{Y}} \stackrel{\text{approx.}}{\sim} \mathcal{N}(0, 1). \quad (6.34)$$

From this result, we can obtain a confidence interval and conduct a hypothesis test for a selected level of statistical significance.

As an example of inference with prediction, we revisit the regression discontinuity design introduced in section 4.3.4. In that study, we estimated the average effect of winning an election on a candidate's wealth in the United Kingdom. Instead of comparing members of Parliament (MPs) who won an election with those who lost it, researchers focused on those who narrowly won or narrowly lost an election. The idea was that if winning an election has a large effect on one's wealth, we should expect a substantial gap in the average wealth at the winning threshold, i.e., the winning margin of zero. Two linear regression models were used to predict the average wealth at this threshold, one based on narrow winners and the other fitted to narrow losers. We reproduce the regression analysis conducted in section 4.3.4, but focus now on just the Tory Party.

The average treatment effect of winning an election results from predicting the average wealth at the winning threshold, i.e., a winning margin of zero. The confidence interval on the predicted value from each regression can be shown by running the `margins` command. Note that as in the `regress` command, the level of statistical significance can be selected by setting the `level()` option to a desired value (the default is 95). We suppress the regression output display by adding the `quietly` command.

```
. use MPs, clear
. * Tory Party at negative margin
. quietly regress lnnet margin if party == "tory" & margin < 0
. margins, at(margin = 0)

Adjusted predictions                               Number of obs     =         121
Model VCE      : OLS
Expression     : Linear prediction, predict()
at            : margin          =         0

+
+-----+
|           Delta-method
|   Margin   Std. Err.      t    P>|t|    [95% Conf. Interval]
+-----+
|   _cons    12.53812   .2141793   58.54   0.000    12.11402    12.96221
+-----+
. * Tory Party at positive margin
. quietly regress lnnet margin if party == "tory" & margin > 0
. margins, at(margin = 0)
```

Adjusted predictions				Number of obs	=	102
Model VCE	:	OLS				
Expression	:	Linear prediction, predict()				
at	:	margin	=	0		
<hr/>						
		Delta-method				
	Margin	Std. Err.	t	P> t	[95% Conf. Interval]	
_cons	13.1878	.1919821	68.69	0.000	12.80691	13.56869

In this output, the predicted value is given by Margin and the lower and upper confidence bands are denoted by the [95% Conf. Interval] columns. For example, the average net wealth for non-MPs at the threshold is estimated to be 12.54 log net wealth with a 95% confidence interval of [12.11, 12.96]. Similarly, the average net wealth for MPs at the threshold is estimated to be 13.19 log net wealth with a 95% confidence interval of [12.81, 13.57].

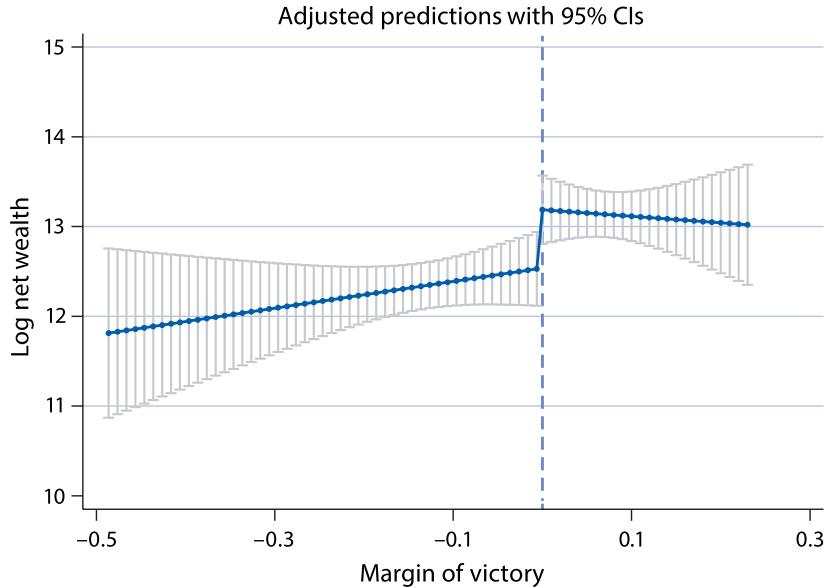
We plot the predicted values obtained from these two regressions (< or > margin), together with their 95% confidence intervals based on the range of predictor  $x$ . We compute the range for each regression using the summarize command to identify both the minimum and maximum values of margin, respectively stored in r(min) and r(max). We then calculate the predictions for each range of electoral margins with 95% confidence intervals by plugging these values into the respective margins commands that we run after each regression.

```
. quietly regress lnnet margin if party == "tory" & margin < 0
. quietly summarize margin if party == "tory" & margin < 0
. quietly margins, at(margin = (`r(min)`(.01)0)) saving(torylow, replace)
.
. quietly regress lnnet margin if party == "tory" & margin > 0
. quietly summarize margin if party == "tory" & margin > 0
. quietly margins, at(margin = (0(.01)`r(max)`)) saving(toryhi, replace)
.
```

Before, we used marginsplot to graph the results. However, here, we want to place two margins plots on the same graph. The combomarginsplot package, which can be installed by typing ssc install combomarginsplot, allows us to combine multiple margins results, provided they are first saved using the saving option. Just like with other graphs, combomarginsplot offers options to tailor the look of the graph. In our example, we change the marker size and lighten the color of the bands marking the confidence intervals. See help combomarginsplot for more features.

```
. combomarginsplot torylow toryhi, xline(0, lpattern(dash)) ///
> legend(off) xlabel(-.5(.2).3) ylabel(10/15) ///
```

```
> xtitle("Margin of victory") ytitle("Log net wealth") ///
> lplot1(msize(tiny)) lcilopts(color(gs12))
.
```



In the plot, we observe that the width of the confidence interval widens as it moves away from the mean value of the predictor. While these two confidence intervals overlap with each other, what we really would like to do is to compute the confidence interval for the difference between these two predicted values. This is because the difference between the two predicted values represents the estimated average treatment effect at the threshold under the regression discontinuity design. Moreover, these two predicted values are assumed to be independent because they are based on two regression models that are fitted to two separate sets of observations. This means that the variance of the difference is the sum of the two variances. To compute the standard error of the difference in the predicted values, we obtain the standard error from each fitted regression. We then use the following formula to compute the standard error of the estimated difference:

$$\text{standard error of } (\hat{Y}_1 - \hat{Y}_0) = \sqrt{(\text{standard error of } \hat{Y}_1)^2 + (\text{standard error of } \hat{Y}_0)^2}.$$

The `estpost` package offers a supplemental command called `estadd` that we place at the start of our `margins` command to save the `margins` output. The results will be stored as e-class results and matrices, which can be viewed by removing `quietly` from our command line or by using `ereturn list`. The standard error and margin, or point estimate, are respectively saved in the `e(margins_se)` and `e(margins_b)` matrices. Recalling the discussion of matrices in chapter 5, we refer to values stored in individual cells of a matrix by row and column. Using `matrix list`, we can observe that both the `e(margins_se)` and `e(margins_b)` matrices store only one value, which we reference with `[1, 1]` (i.e., first row, first column). We save these values as scalars to make additional calculations.

```

. quietly regress lnnet margin if party == "tory" & margin < 0
. quietly estadd margins, at(margin = 0)
. matrix list e(margins_se)
symmetric e(margins_se) [1,1]
    _cons
r1  .21417933

. matrix list e(margins_b)
symmetric e(margins_b) [1,1]
    _cons
r1  12.538116

. * save standard error and point estimate (negative margin)
. scalar se0 = e(margins_se) [1,1]
. scalar fit0 = e(margins_b) [1,1]
.

. quietly regress lnnet margin if party == "tory" & margin > 0
. quietly estadd margins, at(margin = 0)
. * save standard error and point estimate (positive margin)
. scalar sel = e(margins_se) [1,1]
. scalar fit1 = e(margins_b) [1,1]
.

. * standard error prediction
. display sel
.19198207

```

Since in this case the predicted value equals the estimated intercept, the standard error one obtains through the `margins` command is equal to the standard error of the intercept in the regression output.

```

. * s.e. of the intercept is the same as s.e. of the predicted value
. display _se[_cons]
.19198207

```

We can now compute the standard error of the estimated difference in the average log net wealth between MPs and non-MPs at the winning threshold. Using this standard error, we compute the confidence interval and conduct a hypothesis test for which the null hypothesis is that winning an election has zero average effect on candidates' net log wealth.

```

. * standard error
. scalar sediff = sqrt(se0^2 + se1^2)
. display sediff
.28762806

. * point estimate
. scalar diffest = fit1 - fit0
. display diffest
.64968614

. * confidence interval
. display diffest - sediff * invnormal(.975) // lower
.0859455
. display diffest + sediff * invnormal(.975) // upper
1.2134268

. * hypothesis test
. scalar zscore = diffest / sediff
. scalar pvalue = 2 * (1 - normal(abs(zscore)))
. display pvalue
.02389759

```

We find that even though the confidence intervals of the two estimates overlap with each other, the difference between these two estimates is statistically significantly different from zero. Indeed, the average effect of winning an election is estimated to be .65 net log wealth with a 95% confidence interval of [.09, 1.21], which does not contain zero. As a result, the two-sided *p*-value is less than the conventional statistical significance level, 0.05, allowing us to reject the null hypothesis of zero average effect. Thus, our analysis suggests that winning an election had a positive impact on candidates' net wealth. The overlap of the confidence intervals of the two estimates does not necessarily imply that the confidence interval of the difference between the two estimates contains zero.

## 6.4 Summary

In this chapter, we introduced a framework for methods of statistical inference that enables us to quantify the degree of uncertainty regarding our estimates. While we can never know how close our estimates are to the unknown truth, we can evaluate the performance of our estimators using a hypothetically repeated randomization of treatment assignment and/or repeated random sampling. We introduced the concept of **unbiasedness**. An unbiased estimator accurately estimates the parameter of interest on average over a hypothetically repeated data-generating process. Using the law of large numbers, we can also show that some estimators have the property of **consistency**, which implies that as the sample size increases, they converge to the true parameter values.

While unbiasedness is an attractive property, we also need to understand the precision of an estimator given that we can obtain only one realization of the estimator. We use **standard**

**error** to quantify how far our estimator is from the true parameter value on average over a repeated data-generating process. Standard error is an estimate of the standard deviation of the sampling distribution of an estimator. Based on standard errors, we can also construct **confidence intervals**, which will contain the true parameter values with a pre-specified probability, again over a repeated data-generating process. We also showed how to conduct a statistical **hypothesis test** by specifying a null hypothesis and examining whether or not the observed data are consistent with this hypothesis. We applied these inferential methods to the analysis of randomized experiments and sample surveys from earlier in this book.

Finally, we introduced **model-based inference**. We used a linear regression model as a probabilistic generative model, from which the data are assumed to be drawn. Under this setting, we can quantify the uncertainty of our estimated coefficients and predicted values. We showed that the least squares estimates of coefficients are unbiased and derived their standard errors. Using these results, we also explained how to construct confidence intervals and conduct hypothesis tests. Similarly, we showed how to quantify the uncertainty about our predicted values and applied the methodology to the regression discontinuity design introduced in an earlier chapter. These statistical methods play an essential role in our inference because they enable us to separate signals from noise, extracting systematic patterns from data.

## 6.5 Exercises

### 6.5.1 SEX RATIO AND THE PRICE OF AGRICULTURAL CROPS IN CHINA

In this exercise, we consider the effect of a change in the price of agricultural goods whose production and cultivation are dominated by either men or women.<sup>4</sup> Our data come from China, where centrally planned production targets during the Maoist era led to changes in the prices of major staple crops. We focus here on tea, the production and cultivation of which required a large female labor force, as well as orchard fruits, for which the labor force was overwhelmingly male. We use price increases brought on by government policy change in 1979 as a proxy for increases in sex-specific income, and ask the following question: Do changes in sex-specific income alter the incentives for Chinese families to have children of one gender over another? The data file `chinawomen.dta`, contains the variables shown in table 6.4, with each observation representing a particular Chinese county in a given year. Note that `post` is an indicator variable that takes the value 1 in a year following the policy change and 0 in a year before the policy change.

1. We begin by examining sex ratios in the postreform period (i.e., the period after 1979) according to whether or not tea crops were sown in the region. Estimate the mean sex ratio in 1985, which we define as the proportion of male births, separately for tea-producing and non-tea-producing regions. Compute the 95% confidence interval for each estimate by assuming independence across counties within a year. Note that we will maintain this assumption throughout this exercise. Furthermore, compute the difference-in-means between the two regions and its 95% confidence interval. Are sex

<sup>4</sup>This exercise is based on Nancy Qian (2008). “Missing women and the price of tea in China: The effect of sex-specific earnings on sex imbalance.” *Quarterly Journal of Economics*, vol. 123, no. 3, pp. 1251–1285.

**Table 6.4.** Chinese Births and Crops Data.

Variable	Description
admin	unique county identifier
birpop	birth population in a given year
biryrs	year of cohort (birth year)
cashcrop	quantity of cash crops planted in the county
orch	quantity of orchard-type crops planted in the county
teasown	quantity of tea sown in the county
sex	proportion of males in the birth cohort
post	indicator variable for the introduction of price reforms

ratios different across these regions? What assumption is required in order for us to interpret this difference as causal?

2. Repeat the analysis in the previous question for subsequent years, i.e., 1980, 1981, 1982, ..., 1990. Create a graph which plots the difference-in-means estimates and their 95% confidence intervals against years. Give a substantive interpretation of the plot.
3. Next, we compare tea-producing and orchard-producing regions before the policy enactment. Specifically, we examine the sex ratio and the proportion of Han Chinese in 1978. Estimate the mean difference, its standard error, and 95% confidence intervals for each of these measures between the two regions. What do the results imply about the interpretation of the results given in question 1?
4. Repeat the analysis for the sex ratio in the previous question for each year before the reform, i.e., from 1962 until 1978. Create a graph which plots the difference-in-means estimates between the two regions and their 95% confidence intervals against years. Give a substantive interpretation of the plot.
5. We will adopt the difference-in-differences design by comparing the sex ratio in 1978 (right before the reform) with that in 1980 (right after the reform). Focus on a subset of counties that do not have missing observations in these two years. Compute the difference-in-differences estimate and its 95% confidence interval. Note that we assume independence across counties but account for possible dependence across years within each county. Then, the variance of the difference-in-differences estimate is given by

$$\begin{aligned} & \mathbb{V}\{(\bar{Y}_{\text{tea},\text{after}} - \bar{Y}_{\text{tea},\text{before}}) - (\bar{Y}_{\text{orchard},\text{after}} - \bar{Y}_{\text{orchard},\text{before}})\} \\ &= \mathbb{V}(\bar{Y}_{\text{tea},\text{after}} - \bar{Y}_{\text{tea},\text{before}}) + \mathbb{V}(\bar{Y}_{\text{orchard},\text{after}} - \bar{Y}_{\text{orchard},\text{before}}), \end{aligned}$$

where dependence across years is given by

$$\begin{aligned}
 & \mathbb{V}(\bar{Y}_{\text{tea,after}} - \bar{Y}_{\text{tea,before}}) \\
 &= \mathbb{V}(\bar{Y}_{\text{tea,after}}) - 2 \text{Cov}(\bar{Y}_{\text{tea,after}}, \bar{Y}_{\text{tea,before}}) + \mathbb{V}(\bar{Y}_{\text{tea,before}}) \\
 &= \frac{1}{n} \left\{ \mathbb{V}(Y_{\text{tea,after}}) - 2 \text{Cov}(Y_{\text{tea,after}}, Y_{\text{tea,before}}) + \mathbb{V}(Y_{\text{tea,before}}) \right\}.
 \end{aligned}$$

A similar formula can be given for orchard-producing regions. In order to do this exercise and calculate the covariance, the data must be transformed so the before and after observations are in the same row. We discuss the `reshape` command in section 7.3.3. However, for now, we provide the reshaped data in Stata file `reshape_china.dta`. What substantive assumptions does the difference-in-differences design require? Give a substantive interpretation of the results.

### 6.5.2 FILEDRAWER AND PUBLICATION BIAS IN ACADEMIC RESEARCH

The peer review process is the main mechanism through which scientific communities decide whether a research paper should be published in academic journals.<sup>5</sup> By having other scientists evaluate research findings, academic journals hope to maintain the quality of their published articles. However, some have warned that the peer review process may yield undesirable consequences. In particular, the process may result in *publication bias* wherein research papers with statistically significant results are more likely to be published. To make matters worse, by being aware of such a bias in the publication process, researchers may be more likely to report findings that are statistically significant and ignore others. This is called *file drawer bias*.

In this exercise, we will explore these potential problems using data on a subset of experimental studies that were funded by the Time-Sharing Experiments in the Social Sciences (TESS) program. This program is sponsored by the National Science Foundation (NSF). The data set necessary for this exercise can be found in the Stata files `filedrawer.dta` and `published.dta`. The `filedrawer` data set contain information about 221 research projects funded by the TESS program. However, not all of those projects produced a published article. The `published` data include information about 53 published journal articles based on TESS projects. This data set records the number of experimental conditions and outcomes and how many of them are actually reported in the published article. Tables 6.5 and 6.6 present the names and descriptions of the variables from these data sets.

1. We begin by analyzing the data contained in the `filedrawer.dta` file. Create a contingency table for the publication status of papers and the statistical significance of their main findings. Do we observe any distinguishable trend toward the publication of strong results? Provide a substantive discussion.
2. We next examine whether there exists any difference in the publication rate of projects with strong versus weak results as well as with strong versus null results. To do so, first create a variable that takes the value of 1 if a paper was published and 0 if it was not

<sup>5</sup>This exercise is based on the following studies: Annie Franco, Neil Malhotra, and Gabor Simonovits (2014). “Publication bias in the social sciences: Unlocking the file drawer.” *Science*, vol. 345, no. 6203, pp. 1502–1505 and Annie Franco, Neil Malhotra, and Gabor Simonovits (2015). “Underreporting in political science survey experiments: Comparing questionnaires to published results.” *Political Analysis*, vol. 23, no. 2 (Spring), pp. 206–312.

**Table 6.5.** File Drawer and Publication Bias Data I.

Variable	Description
id	study identifier
dv	publication status
iv	statistical significance of the main findings
maxh	H-index (highest among authors)
journal	discipline of the journal for published articles

**Table 6.6.** File Drawer and Publication Bias Data II.

Variable	Description
idp	paper identifier
conds	number of conditions in the study
condp	number of conditions presented in the paper
outs	number of outcome variables in the study
outp	number of outcome variables used in the paper

published. Then, perform two-tailed tests of the difference in the publication rates for the aforementioned comparisons of groups, using 95% as the significance level. Briefly comment on your findings.

3. Using Monte Carlo simulations, derive the distribution of the test statistic under the null hypothesis of no difference for each of the two comparisons you made in the previous question. To run this simulation, it is necessary to first define a program that randomly assigns one of the three paper types (null, weak, or strong results) before calculating the difference-in-means, which should be stored as a return result. Do you attain similar  $p$ -values (for a two-tailed test) to those obtained in the previous question?
4. Conduct the following power analysis for a one-sided hypothesis test where the null hypothesis is that there is no difference in the publication rate between the studies with strong results and those with weak results. The alternative hypothesis is that the studies with strong results are less likely to be published than those with weak results. Use 95% as the significance level and assume that the publication rate for the studies with weak results is the same as the observed publication rate for those studies in the data. How many studies do we need in order to detect a 5 percentage point difference in the publication rate and for the test to attain a power of 95%? For the number of observations in the data, what is the power of the test of differences in the publication rates?

5. The H-index is a measure of the productivity and citation impact of each researcher in terms of publications. More capable researchers may produce stronger results. To shed more light on this issue, conduct a one-sided test for the null hypothesis that the mean H-index is lower or equal for projects with strong results than those with null results. What about the comparison between strong versus weak results? Do your findings threaten those presented for question 2? Briefly explain.
6. Next, we examine the possibility of file drawer bias. To do so, we will use two scatterplots, one that plots the total number of conditions in a study (horizontal axis) against the total number of conditions included in the paper (vertical axis). Make the size of each dot proportional to the number of corresponding studies by setting the [fweight = ] specification. The number of studies can be calculated using the `bysort` command with `egen` and the `sum()` function. The second scatterplot will focus on the number of outcomes in the study (horizontal axis) and the number of outcomes presented in the published paper (vertical axis). As in the previous plot, make sure each circle is weighted by the number of cases in each category. Based on these plots, do you observe problems in terms of underreporting?
7. Create a variable that represents the total number of possible hypotheses to be tested in a paper by multiplying the total number of conditions and outcomes presented in the questionnaires. Suppose that these conditions yield no difference in the outcome. What is the average (per paper) probability that at the 95% significance level we reject at least one null hypothesis? What about the average (per paper) probability that we reject at least two or three null hypotheses? Use the binomial CDF to compute probabilities. Briefly comment on the results.

### 6.5.3 THE 1932 GERMAN ELECTION IN THE WEIMAR REPUBLIC

Who voted for the Nazis? Researchers attempted to answer this question by analyzing aggregate election data from the 1932 German election during the Weimar Republic.<sup>6</sup> We analyze a simplified version of the election outcome data, which records, for each precinct, the number of eligible voters as well as the number of votes for the Nazi party. In addition, the data set contains the aggregate occupation statistics for each precinct. Table 6.7 presents the variable names and descriptions of the data file `nazis.dta`. Each observation represents a German precinct.

The goal of the analysis is to investigate which types of voters (based on their occupation category) cast ballots for the Nazi party in 1932. One hypothesis says that the Nazis received much support from blue-collar workers. Since the data do not directly tell us how many blue-collar workers voted for the Nazis, we must infer this information using a statistical analysis with certain assumptions. Such an analysis, where researchers try to infer individual behaviors from aggregate data, is called *ecological inference*.

To think about ecological inference more carefully in this context, consider the following simplified table for each precinct  $i$ .

<sup>6</sup>This exercise is based on the following article: G. King, O. Rosen, M. Tanner, A. F. Wagner (2008). “Ordinary economic voting behavior in the extraordinary election of Adolf Hitler.” *Journal of Economic History*, vol. 68, pp. 951–996.

**Table 6.7.** 1932 German Election Data.

Variable	Description
shareself	proportion of self-employed potential voters
shareblue	proportion of blue-collar potential voters
sharewhite	proportion of white-collar potential voters
sharedomestic	proportion of domestically employed potential voters
shareunemployed	proportion of unemployed potential voters
nvoter	number of eligible voters
nazivote	number of votes for Nazis

	Occupation		
	Blue-collar	Non-blue-collar	
Vote choice	$W_{i1}$	$W_{i2}$	$Y_i$
	$1 - W_{i1}$	$1 - W_{i2}$	$1 - Y_i$
	$X_i$	$1 - X_i$	

The data at hand tells us only the proportion of blue-collar voters  $X_i$  and the vote share for the Nazis  $Y_i$  in each precinct, but we would like to know the Nazi vote share among the blue-collar voters  $W_{i1}$  and among the non-blue-collar voters  $W_{i2}$ . Then, there is a deterministic relationship between  $X$ ,  $Y$ , and  $\{W_1, W_2\}$ . Indeed, for each precinct  $i$ , we can express the overall Nazi vote share as the weighted average of the Nazi vote share of each occupation:

$$Y_i = X_i W_{i1} + (1 - X_i) W_{i2}. \quad (6.35)$$

1. We exploit the linear relationship between the Nazi vote share  $Y_i$  and the proportion of blue-collar voters  $X_i$  given in equation (6.35) by regressing the former on the latter. That is, fit the following linear regression model:

$$\mathbb{E}(Y_i | X_i) = \alpha + \beta X_i. \quad (6.36)$$

Begin by creating a variable for the Nazi vote share. Next, compute the estimated slope coefficient, its standard error, and the 95% confidence interval. Give a substantive interpretation of each quantity.

2. Based on the fitted regression model from the previous question, predict the average Nazi vote share  $Y_i$  given various proportions of blue-collar voters  $X_i$ . Specifically, using `margins` and `marginsplot` plot the predicted value of  $Y_i$  (the vertical axis) against various values of  $X_i$  within its observed range (the horizontal axis) as a solid line. Add 95% confidence intervals as dashed lines. Give a substantive interpretation of the plot.

3. Fit the following alternative linear regression model:

$$\mathbb{E}(Y_i | X_i) = \alpha^* X_i + (1 - X_i)\beta^*. \quad (6.37)$$

Note that this model does not have an intercept. How should one interpret  $\alpha^*$  and  $\beta^*$ ? How are these parameters related to the linear regression model given in equation (6.36)?

4. Fit a linear regression model where the overall Nazi vote share is regressed on the proportion of each occupation. The model should contain no intercept and five predictors, each representing the proportion of a certain occupation type. Interpret the estimate of each coefficient and its 95% confidence interval. What assumption is necessary to permit your interpretation?
5. We now consider a model-free approach to ecological inference. That is, we ask how much we can learn from the data alone without making an additional modeling assumption. Given the relationship in equation (6.35), for each precinct, obtain the smallest value that is logically possible for  $W_{i1}$  by considering the scenario in which all non-blue-collar voters in precinct  $i$  vote for the Nazis. Express this value as a function of  $X_i$  and  $Y_i$ . Similarly, what is the largest possible value for  $W_{i1}$ ? Calculate these bounds, keeping in mind that the value for  $W_{i1}$  cannot be negative or greater than 1. Finally, compute the bounds for the nationwide proportion of blue-collar voters who voted for the Nazis (i.e., combining the blue-collar voters from all precincts by computing their weighted average based on the number of blue-collar voters. The weighted average can be calculated by adding the [weight = ] option to the summarize command). Give a brief substantive interpretation of the results.

To compute the bounds, we solve for  $W_{i1}$  within each precinct, yielding  $W_{i1} = \frac{(Y_i - (1 - X_i)W_{i2})}{X_i}$ . The minimum and maximum value of  $W_{i1}$  are obtained by setting  $W_{i2} = 1$  (all non-blue-collar voters vote for the Nazis) and  $W_{i2} = 0$  (none of the non-blue-collar voters vote for the Nazis), respectively. We then adjust the minimum and maximum values of  $W_{i1}$  to 0 and 1, respectively, if those estimates are negative or above 1. In summary, the bounds are given by:

$$\max\left(0, \frac{Y_i + X_i - 1}{X_i}\right) \leq W_{i1} \leq \min\left(1, \frac{Y_i}{X_i}\right).$$

# Chapter 7

---

## Discovery

The greatest value of a picture is when it forces us to notice what we never expected to see.

— John W. Tukey, *Exploratory Data Analysis*

Over the past couple of decades, the variety as well as volume of data analyzed in quantitative social science research has dramatically increased. In this chapter, we demonstrate how the coding basics we introduced in the preceding chapters provide a fundamental foundation for more complex analyses, supporting methods at the technical forefront. We introduce three types of data that were not analyzed in previous chapters: network, spatial, and textual data. We conduct *exploratory data analysis* to inductively learn about the underlying patterns and structure of these data. We saw an example of such analysis applied to the degree of political polarization in chapter 3. Here, we first analyze network data, which record the relationships among units. As examples, we will explore the marriage network in Renaissance Florence and social media data from Twitter. Second, we visualize spatial data and examine changes in patterns across time and space, looking specifically at the cholera outbreak in the 19th century and the expansion of Walmart retail stores in the 21st century. Finally, we analyze textual data to discover topics and predict authorship of documents based on the frequency of word usage. Our application is the disputed authorship of *The Federalist Papers*. Unlike in previous chapters, the examples in this chapter are reliant on user-written programs to further showcase Stata's flexibility.<sup>1</sup>

### 7.1 Network Data

*Network data* describes relationships among units rather than units in isolation. Examples include friendship networks among people, citation networks among academic articles, and trade and alliance networks among countries. Analysis of network data differs from the data analyses we have covered so far in that the unit of analysis is a relationship.

<sup>1</sup> All except one example in this chapter will work with Stata/IC.

**Table 7.1.** Florence Marriage Network Data.

FAMILY	ACCIAIUOL	ALBIZZI	...	LAMBERTES	MEDICI	...	STROZZI	TORNABUON
ACCIAIUOL	0	0	...	0	1	...	0	0
ALBIZZI	0	0	...	0	1	...	0	0
⋮			⋮			⋮		
LAMBERTES	0	0	...	0	0	...	0	0
MEDICI	1	1	...	0	0	...	0	1
⋮			⋮			⋮		
STROZZI	0	0	...	0	0	...	0	0
TORNABUON	0	0	...	0	1	...	0	0

The data are in the form of an adjacency matrix where each entry represents whether a family in its row has a marriage relationship with another family in its column.

### 7.1.1 MARRIAGE NETWORK IN RENAISSANCE FLORENCE

We introduce the basic concepts and methods for network data by analyzing a well-known data set about the marriage network in Renaissance Florence.<sup>2</sup> The data file, florentine.dta, contains an *adjacency matrix* whose entries represent the existence of relationships between two units (one unit represented by the row and the other represented by the column). Specifically, there are 16 elite Florentine families in the data, leading to a  $16 \times 16$  adjacency matrix. If the  $(i,j)$  entry of this adjacency matrix is 1, then it implies that the  $i$ th and  $j$ th Florentine families had a marriage relationship. In contrast, a value of 0 indicates the absence of a marriage. Table 7.1 displays part of this data set. We print out the part of the adjacency matrix corresponding to the first five families.

```
. * the first column "family" represents row names
. list family - castellan in 1/5
```

	family	acciaiuol	albizzi	barbad_i	bischeri	castel_n
1.	ACCIAIUOL	0	0	0	0	0
2.	ALBIZZI	0	0	0	0	0
3.	BARBADORI	0	0	0	0	1
4.	BISCHERI	0	0	0	0	0
5.	CASTELLAN	0	0	1	0	0

<sup>2</sup>This section is in part based on John F. Padgett and Christopher K. Ansell (1993). “Robust action and the rise of the Medici, 1400–1434.” *American Journal of Sociology*, vol. 98, no. 6, pp. 1259–1319.

The submatrix shows that there was only one marriage relationship among these five families. The marriage was between the Barbadori and Castellan families. This adjacency matrix represents an *undirected network* because the matrix contains no directionality. We could add directionality by incorporating which family proposed a marriage if such information were available. In contrast, the Twitter data we analyze later are an example of a *directed network* where any relationship between a pair of units specifies a *sender* and a *receiver*. For an undirected network, the adjacency matrix is symmetric: the  $(i, j)$  element has the same value as the  $(j, i)$  element. Using the `egen` command with the `rowtotal` option, we can check which family had the largest number of marriage relationships.

```
. egen marriage = rowtotal(acciaiuol - tornabuon)
. list family marriage
```

	family	marriage
1.	ACCIAIUOL	1
2.	ALBIZZI	3
3.	BARBADORI	2
4.	BISCHERI	3
5.	CASTELLAN	3
6.	GINORI	1
7.	GUADAGNI	4
8.	LAMBERTES	1
9.	MEDICI	6
10.	PAZZI	1
11.	PERUZZI	3
12.	PUCCI	0
13.	RIDOLFI	3
14.	SALVIATI	2
15.	STROZZI	4
16.	TORNABUON	3

The result shows that the Medici family had six marriage relationships. It turns out that through this marriage network, the Medici family made themselves the most powerful faction in Renaissance Florence and eventually took over the state.

Network data carry information about the relationships between units. A **directed network** contains directionality, with senders and receivers, whereas an **undirected network** does not. An **adjacency matrix**, whose entries indicate the existence or

absence of a relationship between two units, is one way to represent network data. An undirected network yields a symmetric adjacency matrix, whereas a directed network does not.

### 7.1.2 UNDIRECTED GRAPH AND CENTRALITY MEASURES

The most common tool for visualizing network data is a *graph*, which is also a mathematical object, as well as a visualization tool. A graph  $\mathcal{G}$  consists of a set of *nodes* (or *vertices*)  $V$  and a set of *edges* (or *ties*)  $E$ , i.e.,  $\mathcal{G} = (V, E)$ . A node represents an individual unit, or a family in our current example, and is typically depicted as a solid circle. An edge, on the other hand, represents the existence of a relationship between any pair of nodes (e.g., a marriage relationship between two families) via a line connecting those nodes.

To visualize network data as a graph, we will use the user-written package **nwcommands**, which offers commands central to network analysis in Stata. Users can find and install the package by typing `net search nwcommands`. Once the **nwcommands-ado** package is installed, typing `nwinstall`, all will download all of the help files, extensions, and dialog box for executing commands. As with other Stata packages, features of **nwcommands** will be readily available for all future sessions after this initial installation. The package is intended for networks of no more than 10,000.

We first declare the data to be network data through the `nwset` command, specifying that our network runs from column (or variable) `acciaiuol` to `tornabuon`. Because we are analyzing an undirected network (i.e., the direction of relationships does not matter), we add the `undirected` option to the end of our command. We do not have to do any further conversions to the data because they are already in matrix format. We can hold multiple network data objects in memory, so we name this one `marriage` within the `name()` option.

```
. * declare network data
. nwset acciaiuol - tornabuon, undirected name(marriage)
```

We can then visualize the marriage network data as a graph using the `nwplot` command, followed by the network name. We use the `label()` option to place the family names, held in the `family` column of our data set, beside each node. The **nwcommands** package has an array of options for further customizing graphs. Some examples include: `size` (to adjust the size of each node), `color` (to adjust the color of each node), `label` (to specify the label of each node), `arrowfactor` (to adjust the size of each edge's arrow), `arcstyle` (to adjust the curvature of the line between each node), and `edgesize` (to adjust the width of each edge). See `help nwplot` for more details.

```
. nwplot marriage, label(family)
Calculating node coordinates...
Plotting network...
```

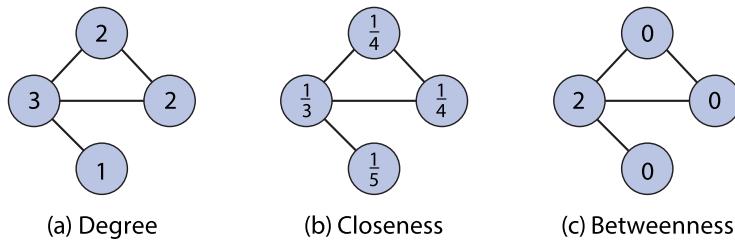
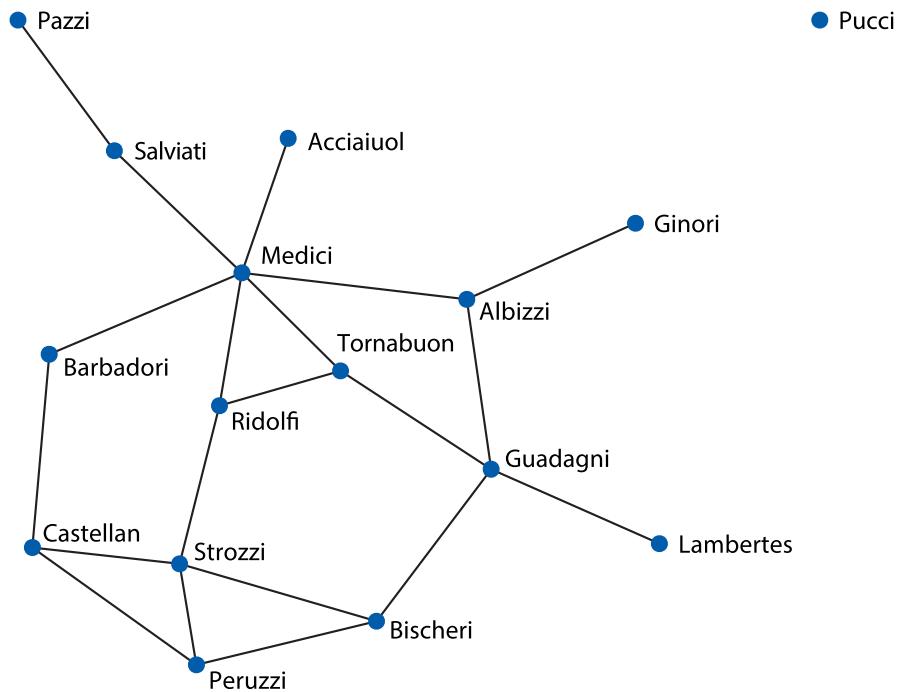


Figure 7.1. Degree, Closeness, and Betweenness in Undirected Network. This simple example of an undirected network illustrates three alternative measures of centrality: degree, closeness, and betweenness.



The Medici family appears to occupy the center of the Florentine marriage network, being connected to various parts of the graph. We now introduce a variety of graph-based measures that can quantify *centrality*, or the extent to which each node is connected to other nodes and appears in the center of a graph. The number of edges, or *degree*, is perhaps the most crude measure of how well a node is connected to the other nodes in a graph. Figure 7.1a illustrates this measure using a simple undirected network example, where degree is indicated as an integer value within each node.

In our example, we found that the Medici family had the largest number of marriage relationships, so it has the highest degree. The degree of every node can be calculated by running the `nwdegree` command. We generate a new variable called `_degree` to store the results.

```
. nwdegree, generate(_degree)
```

---

Network name: marriage

---

Degree distribution

_degree	Freq.	Percent	Cum.
0	1	6.25	6.25
1	4	25.00	31.25
2	2	12.50	43.75
3	6	37.50	81.25
4	2	12.50	93.75
6	1	6.25	100.00
Total	16	100.00	

Degree centralization:: .2666666666666667

```
. * show family and degree columns
. list family _degree
```

	family	_degree
1.	ACCIAIUOL	1
2.	ALBIZZI	3
3.	BARBADORI	2
4.	BISCHERI	3
5.	CASTELLAN	3
6.	GINORI	1
7.	GUADAGNI	4
8.	LAMBERTES	1
9.	MEDICI	6
10.	PAZZI	1
11.	PERUZZI	3
12.	PUCCI	0
13.	RIDOLFI	3
14.	SALVIATI	2
15.	STROZZI	4
16.	TORNABUON	3

Degree is problematically a local measure because it simply counts the number of edges (or lines) that come out of a given node. As a result, it does not account for the structure of the graph beyond its immediate neighbors. As an alternative, we can count the sum of edges from a given node to all other nodes in a graph, including the ones that are not directly connected. This measure, called *farness*, describes how far apart a given node is from the other nodes in the graph. This contrasts with degree, which counts the number of connected nodes. The inverse of farness, *closeness*, represents another measure of centrality. The closeness for node  $v$  is defined as

$$\text{closeness}(v) = \frac{1}{\text{farness}(v)}$$

$$= \frac{1}{\sum_{u \in V, u \neq v} \text{distance between } v \text{ and } u},$$

where the summation is taken over all nodes other than  $v$  itself. The distance between two nodes is the number of edges in the shortest path, which is the shortest sequence of connected nodes, between the two nodes of interest. Figure 7.1b shows the values of this measure for each node in a simple example of an undirected network. We may normalize this measure by multiplying it by the number of nodes in the graph minus 1.

Within the nwcommands package, the nonnormalized measure of closeness is referred to as *nearness* and the normalized measure is called *closeness*. In our analysis, we will focus on the nonnormalized measure when we refer to closeness. The nwcloseness command can compute both measures. But if a node is not connected to any other node, neither nearness nor closeness can be calculated. To calculate both measures in our example, we specify the number of nodes in the graph (i.e., 16). We save the measures in new variables using the generate() option. By default, the results of nwcloseness are returned in the order of closeness, farness, and nearness.

```
. nwcloseness, unconnected(16) generate(close_out far_out near_out)
```

---

```
Network name: marriage
```

---

#### Closeness centrality

Variable	Obs	Mean	Std. Dev.	Min	Max
close_out	16	.2848275	.0692033	.0625	.3658537
far_out	16	62.625	47.71565	41	240
near_out	16	.0189885	.0046136	.0041667	.0243902

```
. list family near_out close_out far_out
```

	family	near_out	close_~t	far_out
1.	ACCIAIUOL	.0185185	.2777778	54
2.	ALBIZZI	.0222222	.3333333	45
3.	BARBADORI	.0208333	.3125	48

4.	BISCHERI	.0196078	.2941177	51
5.	CASTELLAN	.0192308	.2884615	52
6.	GINORI	.0172414	.2586207	58
7.	GUADAGNI	.0217391	.326087	46
8.	LAMBERTES	.0169492	.2542373	59
9.	MEDICI	.0243902	.3658537	41
10.	PAZZI	.0153846	.2307692	65
11.	PERUZZI	.0185185	.2777778	54
12.	PUCCI	.0041667	.0625	240
13.	RIDOLFI	.0227273	.3409091	44
14.	SALVIATI	.0192308	.2884615	52
15.	STROZZI	.0208333	.3125	48
16.	TORNABUON	.0222222	.3333333	45

We see that the Pucci family's nearness, for example, is equal to  $1/(15 \times 16) = 0.004$ . As with degree, we find that the Medici family has the largest value of closeness. To facilitate the interpretation of this measure, we can calculate the average number of edges from a given node to another node. This is done by dividing the farness by the number of other nodes on a graph. In the current example, we have a total of 16 nodes, so we divide the farness by 15. The results imply that, on average, there are 2.7 edges between the Medici family and another family in this network, which is the lowest among all families considered in this network data.

```
. generate avg_edge = 1 / (near_out * 15)
. list family avg_edge
```

	family	avg_edge
1.	ACCIAIUOL	3.6
2.	ALBIZZI	3
3.	BARBADORI	3.2
4.	BISCHERI	3.4
5.	CASTELLAN	3.466666
6.	GINORI	3.866667
7.	GUADAGNI	3.066667
8.	LAMBERTES	3.933333
9.	MEDICI	2.733333
10.	PAZZI	4.333333

11.	PERUZZI	3.6
12.	PUCCI	16
13.	RIDOLFI	2.933333
14.	SALVIATI	3.466666
15.	STROZZI	3.2
<hr/>		
16.	TORNABUON	3

A different type of centrality measure is *betweenness*. According to this measure, a node is considered to be central if it is responsible for connecting other nodes. In particular, if we assume that communication between a pair of nodes occurs through the shortest path between them, a node that lies on many such shortest paths may possess special leverage within a network. For a given node  $v$ , we calculate betweenness in three steps. First, compute the proportion of the shortest paths between two other nodes,  $t$  and  $u$ , that contain  $v$ . For example, two shortest paths occur between the Albizzi family and Tornabuon family, but we want only the proportion that contain  $v$ . Second, calculate this proportion for every unique pair of nodes  $t$  and  $u$  in the graph, excluding  $v$ . Third, sum all proportions. The formal definition is given by

$$\text{betweenness}(v) = \sum_{(t,u) \in V, t \neq v, u \neq v} \frac{\text{number of shortest paths that contain node } v}{\text{number of shortest paths between nodes } t \text{ and } u}.$$

Figure 7.1c illustrates this centrality measure in the same undirected network example used for the other two measures.

The `nwbetween` command provides our measure of betweenness, which we save as `between`. The `nosym` option requests that we do not symmetrize our data before calculating the shortest paths. The default is to symmetrize the data, which we would do if we wanted to transform directed network data into undirected network data.

```
. nwbetween, generate(between) nosym
Network name: marriage
Betweenness centrality
Variable | Obs Mean Std. Dev. Min Max
between | 16 9.75 12.30056 0 47.5
. list family between
```

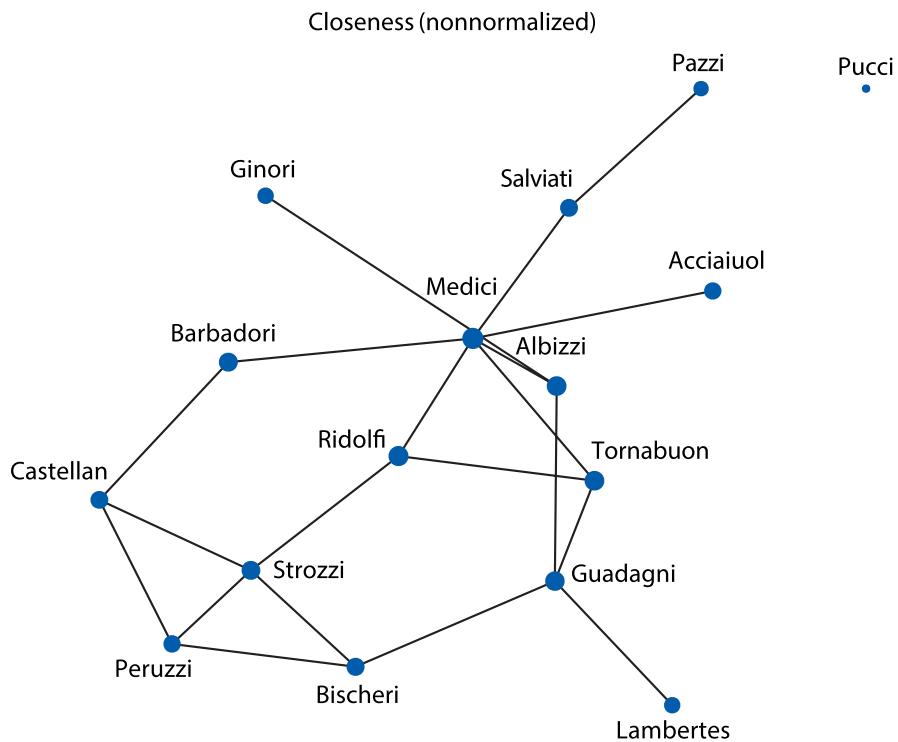
	family	between
1.	ACCIAIUOL	0
2.	ALBIZZI	19.33333
3.	BARBADORI	8.5
4.	BISCHERI	9.5
5.	CASTELLAN	5
6.	GINORI	0
7.	GUADAGNI	23.16667
8.	LAMBERTES	0
9.	MEDICI	47.5
10.	PAZZI	0
11.	PERUZZI	2
12.	PUCCI	0
13.	RIDOLFI	10.33333
14.	SALVIATI	13
15.	STROZZI	9.333333
16.	TORNABUON	8.333333

We find that by far, the Medici family has the highest value of betweenness. In fact, since any given node can be uniquely paired with 105 other nodes, the Medici family lies in the shortest path of approximately 45% of all possible pairs of other nodes.

A **graph** is another way to represent network data where nodes (vertices) represent units and an edge (or tie) between two nodes indicates that a relationship exists between them. There are various **centrality** measures, including degree, closeness, and betweenness. These measures evaluate the extent to which each node plays a central role in a graph.

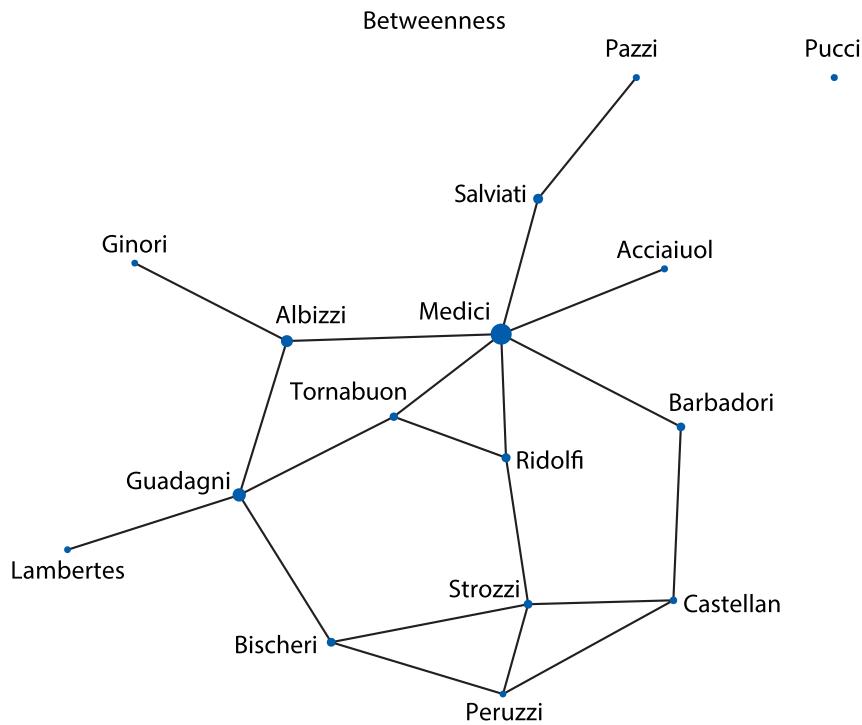
We visualize the Florentine marriage network data by making the size of each node proportional to two centrality measures, closeness and betweenness. Again, we use the non-normalized measure of closeness, which is referred to as nearness in `nwcommands`. Within the `size()` option, we specify the variable on which the node size should be based. We again include family names as labels, but also take advantage of additional customizations offered by `nwplot`. For greater readability, we use `mlabposition()` and `mlabgap()` to respectively place the labels above the nodes and increase the gap in between. We also increase the space between the title and graph itself using `margin` within our title specification.

```
. nwplot marriage, size(near_out) ///
>     label(family) title("Closeness (nonnormalized)", margin(b = 4)) ///
>     labelopt(mlabposition(12) mlabgap(2.0)) ///
>     legend(off) name(near, replace)
Calculating node coordinates...
Plotting network...
```



```
. nwplot marriage, size(between) ///
>     label(family) title("Betweenness", margin(b = 4)) ///
>     labelopt(mlabposition(12) mlabgap(2.0)) ///
>     legend(off) name(between, replace)
Calculating node coordinates...
Plotting network...
```

The graphs illustrate that the Medici family stands out especially in terms of betweenness, while the closeness measure suggests they are one of several well-connected families. In sum, using three measures of centrality—degree, closeness, and betweenness—we find that the Medici family is the most connected and central in the network of Florentine marriage relationships. In Renaissance Florence, the Medici family had the largest number of marriage



relationships, was closely connected to other families, and occupied a critical position in marriages among other families.

### 7.1.3 TWITTER FOLLOWING NETWORK

The Florentine marriage network data exemplify an *undirected network* where each edge has no directionality. Next, we analyze Twitter-following data among US senators as *directed network* data. In this data set, an edge represents an instance of a senator following another senator's Twitter account.<sup>3</sup> The data consist of two files, one listing pairs of the Twitter screen names of the following and followed politicians, `twitter-following.dta`, and the other containing information about each politician, `twitter-senator.dta`. Table 7.2 lists the names and descriptions of variables in these two data files.

We begin by creating two adjacency matrices with the `twitter-following` data. For directed network data, the  $(i, j)$ th element of the adjacency matrix is 1 if an edge connects node  $i$  to node  $j$ . A value of 0 indicates the absence of any relationship. Consequently, unlike the case of undirected network data, the adjacency matrix is asymmetric: the  $(i, j)$ th element of this matrix may not equal its  $(j, i)$ th element. As before, we declare our data to be network data and also create the adjacency matrices by using the `nwset` command. Each observation in the data indicates whether there is a tie between the sender and the receiver (also called the source and the target). In the command, the first variable listed is the sender (or *from*) node, while the second variable listed is the receiver (or *to*) node. To reflect that our data are structured in this way by two columns (*from* and *to*), we add the `edgelist` option when making our `nwset` declaration.

<sup>3</sup>This data set is generously provided by Pablo Barberá.

**Table 7.2.** Twitter Following Data.

Variable	Description
Twitter-following data	
following	Twitter screen name of the following senator
followed	Twitter screen name of the followed senator
Twitter senator data	
screen_name	Twitter screen name
name	name of senator
party	party (D = Democrat, R = Republican, I = Independent)
state	state abbreviation

The data are in two files, one listing the pairs of following and followed senators and the other containing information about each senator.

We declare a directed network using the `twitter-following` data set, specifying an asymmetric relationship from the `following` to the `followed` senators. We name this network `twitter`, saving the data as a temporary file to merge with the senator data and we refer to it again later.

```

. * create directed network
. use twitter-following, clear

. nwset following followed, edgelist name(twitter)

. tempfile twitter

. * merge with senator data
. use twitter-senator, clear

. rename screen_name _nodelab

. merge 1:1 _nodelab using `twitter'

      Result                      # of obs.

not matched                               0
matched                                91  (_merge==3)

. save `twitter', replace

```

#### 7.1.4 DIRECTED GRAPH AND CENTRALITY

We can define the three centrality measures discussed earlier for a directed network. We now have two types of *degree* measures. The sum of edges coming to a node (i.e., the number

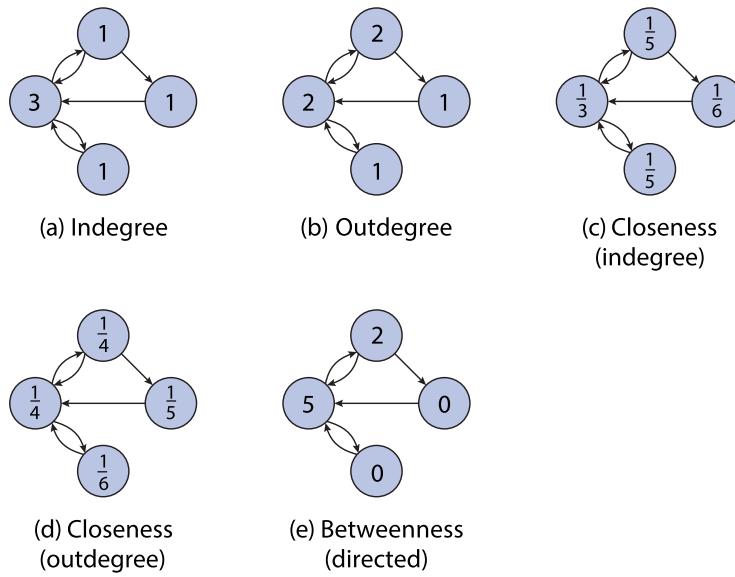


Figure 7.2. Degree, Closeness, and Betweenness in a Directed Network. This simple example of directed network data illustrates three alternative measures of centrality: degree, closeness, and betweenness.

of times a politician's Twitter account is followed by another politician) is called *indegree*, whereas the sum of edges coming out of a node (i.e., the number of times a politician follows the Twitter account of another politician) is called *outdegree*. Figures 7.2a and 7.2b illustrate the two degree measures using a simple directed network. The nwdegree command calculates the indegree and outdegree for directed networks, by default. We compute and store `_indegree` and `_outdegree` as additional variables. We suppress the tables of indegree and outdegree values by adding `nofreq` for both the `in()` and `out()` options. If these are not specified, the nwdegree command creates a frequency table of indegree and of outdegree.

```
. nwdegree twitter, generate(degree) in(nofreq) out(nofreq)
```

---

```
Network name: twitter
```

---

```
Degree distribution
```

```
Indegree centralization:: .2425925925925926
```

```
Outdegree centralization:: .5234567901234568
```

Next, we list the cases with the three greatest values of indegree and outdegree. To do this, we use the gsort command, which allows both ascending and descending sorts. We place a negative sign in front of a variable when we want it sorted in descending order. We

sort politicians by descending order of `_indegree` and list the first three senators. We repeat the process for the greatest values of `_outdegree`. When two or more variables are listed in a `gsort` command, Stata will initially sort observations by the variable listed first. Observations will only be resorted based on the second variable if there are multiple values of the first. For example, one may first sort years and then months and days within those years.

```
. gsort -_indegree -_outdegree
. list _nodelab name party state _indegree _outdegree in 1/3
```

	<code>_nodelab</code>	<code>name</code>	<code>party</code>	<code>state</code>	<code>_indeg~e</code>	<code>_outde~e</code>
1.	SenJohnMcCain	John McCain	R	AZ	64	15
2.	lisamurkowski	Lisa Murkowski	R	AK	60	87
3.	SenatorCollins	Susan M. Collins	R	ME	58	79

```
. gsort -_outdegree -_indegree
. list _nodelab name party state _indegree _outdegree in 1/3
```

	<code>_nodelab</code>	<code>name</code>	<code>party</code>	<code>state</code>	<code>_indeg~e</code>	<code>_outde~e</code>
1.	SenDeanHeller	Dean Heller	R	NV	55	89
2.	SenBobCasey	Robert P. Casey, Jr.	D	PA	43	88
3.	SenatorTimScott	Tim Scott	R	SC	41	88

The other two measures of centrality we introduced, *closeness* and *betweenness*, can be defined for directed network data as well. There are three ways to define a path from one node to another. We can ignore directionality as in the case of undirected networks or incorporate it in one of two ways: traveling along an outgoing path in its direction, or traveling along an incoming path against its direction. Closeness for incoming paths corresponds to `indegree`, while closeness for outgoing paths corresponds to `outdegree`. Figures 7.2c and 7.2d illustrate the closeness measures based on incoming and outgoing paths, respectively.

In Stata, `nwcommands` only calculates closeness for outgoing paths. To examine closeness for incoming paths, we create a network that reverses the relationships, where the followed follow the following in our data set, naming this network `reverse_twitter`. We then reload our temporary `twitter` file to access the merged senator data. In the `nwcloseness` commands, we again request that the paths are not made symmetric before calculation by adding the `nosym` option. Because there are unconnected nodes, we set the length of

these paths to the number of nodes, or 91, using the `unconnected()` option. We generate variables for closeness, farness, and nearness separately for the outgoing and incoming paths.

```
. * create reverse network
. use twitter-following, clear

. nwset followed following, edgelist name(reverse_twitter) directed
. use `twitter`, clear
.

. * calculate closeness measures
. nwcloseness twitter, nosym unconnected(91) generate(close_out far_out
near_out)

Network name: twitter

Closeness centrality

Variable | Obs      Mean   Std. Dev.    Min     Max
-----+-----+-----+-----+-----+-----+
close_out | 91      .6497337  .2110816  .010989  .989011
far_out   | 91      579.4945   1845.445    91      8190
near_out  | 91      .0072193   .0023454  .0001221  .010989

. nwcloseness reverse_twitter, nosym unconnected(91) generate(close_in far_in
near_in)

Network name: reverse_twitter

Closeness centrality

Variable | Obs      Mean   Std. Dev.    Min     Max
-----+-----+-----+-----+-----+-----+
close_in | 91      .1555063  .0058746  .1453958  .1832994
far_in   | 91      579.4945   19.85132   491      619
near_in  | 91      .0017278  .0000653  .0016155  .0020367
```

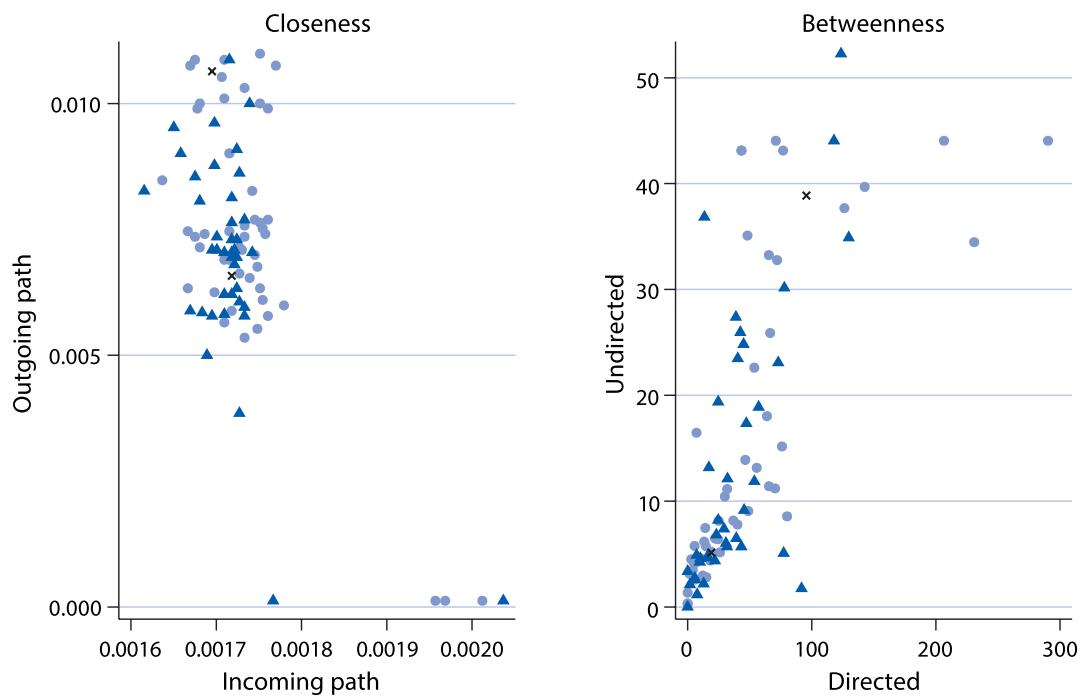
Betweenness only offers the opportunity to explore the difference between directed and undirected networks in this case because the distinction between incoming and outgoing paths does not make sense from the perspective of a node in the path between two other nodes (see figure 7.2e). We run the `nwbetween` command twice, once without the `nosym` option and the other with, to specify whether or not to consider directionality. In the first betweenness calculation, we symmetrize the network, essentially creating an undirected measure. In the second calculation, we do not symmetrize the network.

```
. nwbetween twitter, generate(bw_dir)
_____
Network name: twitter
_____
Betweenness centrality
Variable | Obs Mean Std. Dev. Min Max
_____
bw_dir | 91 44.43956 49.77678 0 290.3766
. nwbetween twitter, generate(bw_undir) nosym
_____
Network name: twitter
_____
Betweenness centrality
Variable | Obs Mean Std. Dev. Min Max
_____
bw_undir | 91 14.71429 14.08407 0 52.27727
```

We now graphically compare the two (nonnormalized) closeness measures, for the incoming versus outgoing paths, and then compare directed betweenness against undirected betweenness in a separate graph. In the plots, we use blue triangles for Democrats, red circles for Republicans, and black crosses for Independents. See page C4 for the full-color versions.

```
. scatter near_out near_in if party == "R", msymbol(O) mcolor(red) || ///
> scatter near_out near_in if party == "D", msymbol(T) mcolor(blue) || ///
> scatter near_out near_in if party == "I", msymbol(X) mcolor(black) ///
> xtitle("Incoming path") ytitle("Outgoing path") title("Closeness") ///
> legend(off) name(near, replace)
. scatter bw_undir bw_dir if party == "R", msymbol(O) mcolor(red) || ///
> scatter bw_undir bw_dir if party == "D", msymbol(T) mcolor(blue) || ///
> scatter bw_undir bw_dir if party == "I", msymbol(X) mcolor(black) ///
> xtitle("Directed") ytitle("Undirected") title("Betweenness") ///
> legend(off) name(btw, replace)
. graph combine near btw
```

There is little association between the two closeness measures based on incoming and outgoing paths. This suggests that in this Twitter network, a senator's closeness to other senators in terms of being followed by them has little relationship to closeness based on the same senator following them. Interestingly, however, the betweenness measure is quite similar, regardless of whether one incorporates directionality. The two betweenness measures suggest that



several Republican senators are well connected and central to the network (see the upper-right corner of the right-hand plot).

As a final alternative measure of centrality, we introduce *PageRank*. PageRank was developed by the cofounders of Google, Sergey Brin and Larry Page, to optimize the ranking of websites for their search engine outcomes. PageRank is computed using an iterative algorithm. In section 3.7, we saw *k*-means clustering as an example of an iterative algorithm. PageRank is based on the idea that nodes with a greater number of incoming edges are more important. Intuitively, we can think of incoming edges as votes of support. In the Twitter example, those senators who have a large number of followers are seen as more important. Furthermore, if a node has an incoming edge from another node with a large number of incoming edges, it results in a greater value of PageRank than if it has an incoming edge from a node with fewer incoming edges. In other words, if the Twitter account of a politician is followed by another politician whose account has many followers, she receives a larger PageRank than she would if followed by a politician with fewer followers. Finally, we note that the sum of PageRank values across all nodes equals 1.

The algorithm begins by assigning a set of initial PageRank values to all nodes. At each iteration, the PageRank value for node  $j$  will be updated using the following equation,

$$\text{PageRank}_j = \frac{1-d}{n} + d \times \sum_{i=1}^n \underbrace{\frac{A_{ij} \times \text{PageRank}_i}{\text{outdegree}_i}}_{\text{"vote" from node } i \text{ to node } j}. \quad (7.1)$$

In this equation,  $A_{ij}$  is the  $(i,j)$ th element of the adjacency matrix indicating whether or not an edge connects node  $i$  to node  $j$ ,  $d$  is a constant to be specified (typically set to 0.85), and  $n$  is the number of nodes. The equation shows that the PageRank for a given node  $j$  equals the sum

of “votes” from other nodes that have an incoming edge into node  $j$ . If there is no edge from node  $i$  to node  $j$ , then  $A_{ij} = 0$  and therefore no vote is given to node  $j$  from node  $i$ . However, if  $A_{ij} = 1$ , then a vote from node  $i$  to node  $j$  is equal to the PageRank value of node  $i$  divided by node  $i$ ’s outdegree. This means that each node must equally allocate its PageRank value across all other nodes to which it has outgoing edges. For example, if a node has a PageRank value of 0.1 and has two outgoing edges, then each receiver obtains 0.05 from this node. This iterative algorithm stops when the PageRank values for all nodes no longer change.

There are several **centrality measures** for **directed networks**, including indegree and outdegree, closeness (based on incoming edges, outgoing edges, or both), and betweenness (with or without directionality). **PageRank** is an iterative algorithm that produces a centrality measure where each node equally allocates its “votes” to other connected nodes.

Because there is no ready-made **PageRank** package available in Stata, we have to define our own program. But we first write code that updates the PageRank at one iteration according to equation (7.1) to better understand the algorithm. In the pseudocode that follows, let  $n$  be the number of nodes in a graph,  $A$  be a network data set, ‘ $d$ ’ be a constant, and  $_pr$  be a variable of PageRank values from the previous iteration. We get the number of nodes from a summary of the network, stored in  $r(nodes)$ , and create a second PageRank variable ( $_prpre$ ).

```
nwsummarize A
local n = r(nodes)
nwdegree A, generate(prdeg) valued
generate _pr = 1 / `n' if _n <= `n'
generate _prpre = _pr
forvalues i =1/`n' {
    generate n`i' = net`i' *_prpre / _outprdeg
    sum n`i'
    replace _pr = (1 - `d') / r(N) + `d' * r(sum) in `i'
    drop n`i'
}
```

We add additional code so the algorithm knows when to stop. We use a `while ()` loop so that the algorithm stops when the differences in PageRank values between two successive iterations become negligible. The syntax for the `while ()` loop is similar to that of other loop types we have previously reviewed (see also section 4.1.2).

```
while condition {
    command1
    command2
    ...
}
```

```
    commandN
}
```

The loop contents are executed repeatedly so long as the conditional statement, `condition`, is evaluated to be TRUE. In our application, we will compute the maximum absolute difference in PageRank values between two successive iterations and stop the algorithm when this becomes less than a prespecified threshold.

To implement the PageRank algorithm, we create the starting values in the program but allow users to specify the constant  $d$  in the syntax (we choose 0.85). We then use the `while()` loop to iteratively run the algorithm until a convergence criterion is satisfied. For the convergence criteria, we use 0.001 as the threshold for the maximal absolute difference in the PageRank values between two successive iterations. We use equal PageRank values across nodes as their starting values.

```
. program pagerank
1. quietly {
2.     syntax anything, [d(real .85)]
3.     tempvar _prpre _dif
4.     nwsummarize `anything'
5.     local n = r(nodes)
6.     nwdegree `anything', generate(prdeg) valued
7.     generate _pr = 1 / `n' if _n <= `n'
8.     generate `_prpre' = _pr
9.     scalar diff = 1
10.    while scalar(diff) > .001 {
11.        replace `_prpre' = _pr
12.        forvalues i =1/`n' {
13.            generate _ndeg = net`i' * `_prpre' / _outprdeg
14.            summarize _ndeg
15.            replace _pr = (1 - `d') / r(N) + `d' * r(sum) in `i'
16.            drop _ndeg
17.        }
18.        summarize _pr
19.        replace _pr = _pr / r(sum)
20.        generate `_dif' = abs(_pr - `_prpre')
21.        summarize `_dif'
22.        scalar diff = r(max)
23.        drop `_dif'
24.    }
25. }
26. end
```

To test this script, we first construct an adjacency matrix with arbitrary values. In chapter 6, we used the `runiformint` command to generate random integers. By adding the `char` specification, we can also generate random letters. We start at 65, which is the byte value for

uppercase letter *A*, through 70 (uppercase letter *F*). The byte value for lowercase *a* is 97. In this exercise, we can think of these letters as representing different websites.

```
. clear
. set seed 12345
. set obs 10
number of observations (_N) was 0, now 10
. * create and view adjacency matrix
. generate name1 = char(runiformint(65, 70))
. generate name2 = char(runiformint(65, 70))
. list name1 name2
```

	name1	name2
1.	C	A
2.	D	A
3.	F	D
4.	E	E
5.	E	D
6.	B	C
7.	A	A
8.	F	A
9.	D	C
10.	B	A

```
. * declare network named "websites" and run PageRank
. nwset name1 name2, edgelist directed name(websites)
. pagerank websites
. list _nodelab _pr if _nodelab!=""
```

	_nodelab	_pr
1.	A	.7934062
2.	B	.025
3.	C	.05875
4.	D	.0542344
5.	E	.0436094
6.	F	.025

The result shows that website *A* has the highest PageRank value. This makes sense because the adjacency matrix shows us that it is the observation with the greatest number of incoming edges, represented by the second column.

We can now apply our pagerank program to the Twitter network data to identify the most influential senators. The output variable holding the page rank value is again called `_pr`. We reload our temporary file created earlier. If this file is no longer in memory, it will be necessary to again reopen the `twitter-following` data, declare its network, and merge it with the senator data. As input, PageRank requires the network name we declared in our original `nwset` command.

```
. use `twitter', clear
. pagerank twitter
```

We visualize the Twitter following network by setting node size proportional to the PageRank variable `_pr`. We remove arrowheads from connecting lines (or edges) by specifying `arrowfactor(0)`. We change the color of the nodes and connecting lines with the `color()` and `lineopt()` options. Nodes are colored by party. The displayed plot uses light blue for Republicans and dark blue for Democrats. The full-color version is on page C4.

```
. nwplot twitter, size(_pr) arrowfactor(0) ///
>   color(party, colorpalette(blue white red)) ///
>   lineopt(lcolor(gs12)) legend(off)
Calculating node coordinates...
Generating splines...
Plotting network...
```

The plot shows that this Twitter network is quite dense but also polarized. Republican senators appear to be clustered together while Democrats form another cluster, suggesting that senators tend to follow other senators of their own party. In addition, while Republican senators appear to have slightly greater PageRank values than Democrats, the partisan difference is minor.

## 7.2 Spatial Data

In addition to networks, we introduce another type of data, *spatial data*. Spatial data are best analyzed by visualization through *maps*. This chapter covers two types of spatial data. One is *spatial point data*, which can be plotted as a set of points on a map. The other is *spatial polygon data*, which represent a sequence of connected points on a map corresponding to the boundaries of certain areas such as counties, districts, and provinces. We also consider *spatial-temporal data*, which are a set of spatial point or polygon data recorded over time, revealing changes in spatial patterns.

### 7.2.1 THE 1854 CHOLERA OUTBREAK IN LONDON

In his book, *Mode of Communication of Cholera*, a British physician John Snow demonstrated the effective use of maps for visualizing the spatial distribution of fatal cholera cases. Snow collected the spatial point data about fatal cases in the Soho neighborhood of London during the 1854 outbreak and plotted this information on a map. Figure 7.3 reproduces the original map. Black rectangle areas, clustered around the center of the map, indicate fatal cholera cases. All water pumps are also indicated by solid circles and labeled as such on the map.

From this map, Snow discovered that fatal cholera cases were clustered around the Broad Street water pump. He speculated that cholera was spread by sewage-contaminated water, a theory the authorities and the water company were reluctant to believe. After extensive research that included close inspection of water and interviews with local residents, Snow concluded that the water pump at the corner of Broad and Cambridge Streets was the source of the cholera outbreak. He concluded by writing,

The result of the inquiry then was, that there had been no particular outbreak or increase of cholera, in this part of London, except among the persons who were in the habit of drinking the water of the above-mentioned pump-well. (p. 40)

Snow also employed a “grand natural experiment” to show that the water supply of the Southwark and Vauxhall Company was responsible for the spread of cholera in London. Figure 7.4 reproduces the spatial polygon map that Snow used to visualize the area of the *natural experiment*, meaning a situation in the world that resembles an experiment without intervention from researchers. The map shows that the Lambeth Company supplied cleaner water to the area further south (indicated by the red region), whereas the Southwark and Vauxhall Company provided contaminated water to the neighborhoods along the River Thames (indicated by the blue region). See page C5 for the full-color version. Snow argued that the overlapping area represented a natural experiment where two companies competed for customers: some people received their water supply from one company while their neighbors received water from the other company. Assuming that the two groups of customers

Figure 7.3. John Snow's Map of Fatal Cholera Cases in London. Black rectangle areas indicate fatal cholera cases, which were found to cluster around the Broad Street water pump. All water pumps are also indicated on the map. *Original Source:* John Snow (1855). *Mode of Communication of Cholera.* London: John Churchill, New Burlington Street.

were alike in all other respects, any difference in their cholera rates resulted from the choice of company.

After much research, Snow concluded that probably no *confounder* affected this *natural experiment*. Based on the discussion in section 2.5.2, confounding factors in this context refer to the variables associated with water companies and cholera outbreak rates of a neighborhood. He describes this experiment succinctly as follows:

The mixing of the supply is of the most intimate kind. The pipes of each Company go down all the streets, and into nearly all the courts and alleys. A few houses are supplied by one Company and a few by the other, according to the decision of the owner or occupier at that time when the Water Companies were in active competition. In many cases a single house has a supply different from that on either side. Each Company supplies both rich and poor, both large houses and small; there is no

Figure 7.4. John Snow's Map of the Natural Experiment. The map shows the area of the natural experiment where two water companies (the Lambeth Company, and the Southwark and Vauxhall Company) compete for customers. This area is represented by the overlap of red (Lambeth) and blue (Southwark and Vauxhall) regions.

difference either in the condition or occupation of the persons receiving the water of the different Companies....

The experiment, too, was on the grandest scale. No fewer than three hundred thousand people of both sexes, of every age and occupation, and of every rank and station, from gentlefolks down to the very poor, were divided into two groups without their choice, and, in most cases, without their knowledge; one group being supplied with water containing the sewage of London, and amongst it, whatever might have come from the cholera patients, the other group having water quite free from such impurity. (pp. 74–75)

By matching the addresses of persons dying of cholera with the companies that supplied them water, Snow was able to show that the overwhelming majority of deaths had occurred in the households with water supplied by the Southwark and Vauxhall Company.

Snow's book illustrates the power of spatial data analysis. In particular, the visualization of spatial data through maps enables researchers to discover previously unknown patterns and present their findings in a convincing manner.

### 7.2.2 SPATIAL DATA IN STATA

In chapter 4, we analyzed the 2008 US presidential election. Figure 4.1 presents a map of the Electoral College, efficiently visualizing the outcome of the election. This is an example of *spatial polygon data*, where each state represents a polygon whose boundaries can be constructed by connecting a series of points. We can then color each polygon or state blue (red) if Barack Obama (John McCain) won the plurality of votes within that state.

In this chapter, we will be using the user-written packages **spmap** and **shp2dta**, available through the Statistical Software Components (SSC) archive (type `ssc install spmap` and `ssc install shp2dta`). The **spmap** package provides basic functionalities and mapping tools that allow us to visualize our spatial data. It should be noted that Stata 15 introduced a new mapping command called **grmap**. It is recommended that individuals working on more recent versions of Stata consult this package (see `help grmap`). Similarly, Stata 15 introduced **spshape2dta**, which is the updated counterpart to **shp2dta**. While **spmap** and **grmap** share overlapping features, **grmap** offers some advances, including those related to the creation of choropleth maps that vary color gradation based on variable values. Both tools, however, require spatial data.

Spatial data are typically stored in what are called shapefiles, which will include the longitude and latitude of geometric locations along with geographic attributes, ranging from details regarding terrain (e.g., mountain elevation and bodies of water) to road networks. Although beyond the scope of this book, shapefiles are a form of vector data, where attributes are represented by lines, points, or polygons. In Stata, we first need to find a shapefile and then convert that vector data into Stata format, which we can do using **shp2dta**. The web provides a wide selection of spatial databases where users can download shapefiles for free. In the United States, the Census Bureau is just one resource. We acquire the files used in our analysis from <https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html>. Navigate to the year 2017 and download the ZIP file `cb_2017_us_state_500k.zip`, which contains data with a cartographic resolution (scale of 500k).

A shapefile is not any one file, but a collection of files. After the file is downloaded and unzipped, one can see several files with different extensions, including `.cpg`, `.dbf`, `.prj`, `.shp`, `.xml`, and `.shx`. Someone working with vector data in geographic information systems (GIS) software would likely use all of these files. However, in Stata, we only need the files with the `.dbf` (dBASE table) and `.shp` (main file) extensions. We run the `shp2dta` command to convert the spatial data to Stata format. Be sure that the files are in the current working directory. Following the `shp2dta` command, we simply add the main file name that precedes each extension (the name should be the same for each—here, it is `cb_2017_us_state_500k`). The `shp2dta` command then has two required options, `database()` and `coord()`, where we specify what we would like to name the data sets converted from the `.dbf` and `.shp` files, respectively. The `.dbf` file, in this case, contains information about each state, while the `.shp` file holds the coordinates for each state. We also specify that the two data files should be linked by the `id` variable created with `genid()`.

```
. shp2dta using cb_2017_us_state_500k, database(usdb) ///
>     coordinates(uscoord) genid(id) replace
type: 5
```

Once the data have been converted, we can open them like any other Stata file. For our analysis, we use the `inlist()` function to drop territories outside of the continental United States. Though not previously introduced, we also use the `regexm()` function to remove any observations with a string match to “Mariana.” We save our cleaned data.

```
. use usdb, clear
. drop if inlist(NAME, "Guam", "Alaska", "Hawaii", ///
>     "Puerto Rico", "American Samoa", ///
>     "United States Virgin Islands") | regexm(NAME, "Mariana")
(7 observations deleted)
. save usdb, replace
file usdb.dta saved
```

We will merge our newly created `usdb.dta` file with other Stata data sets to draw different maps. First, the data file `uscities.dta` contains information on cities in the United States, including city name (stored under the variable called `name`), state (`countryetc`), population (`pop`), latitude (`lat`), longitude (`long`), and an indicator for whether the city is the capital of the country (`capital = 1`), capital of a state (`capital = 2`), or neither (`capital = 0`). We list the first five observations in this data set. Because we need a shared identifier to complete the merge, we create a new variable called `STUSPS`, which can be found in the `usdb` data set and holds the state abbreviation. The state abbreviation in the `uscities` file is included as the last two string characters in the `name` variable. Consequently, we tell Stata to create a new variable using the `substr()` function, where we specify that we want a character string, based on variable `name`, that starts with the second to last letter (-2) and has a length of 2. With a key identifier in place, we merge the `uscities` and `usdb` data, keeping only matched observations.

```
. use uscities, clear
. list in 1/5
```

	v1	name	countr~c	pop	lat	v6	capital
1.	1	Abilene TX	TX	113888	32.45	-99.74	0
2.	2	Akron OH	OH	206634	41.08	-81.52	0
3.	3	Alameda CA	CA	70069	37.77	-122.26	0
4.	4	Albany GA	GA	75510	31.58	-84.18	0
5.	5	Albany NY	NY	93576	42.67	-73.8	2

```
. generate STUSPS = substr(name, -2, 2)
. merge m:1 STUSPS using usdb
```

Result	# of obs.
not matched	4
from master	4 (_merge==1)
from using	0 (_merge==2)
matched	1,001 (_merge==3)

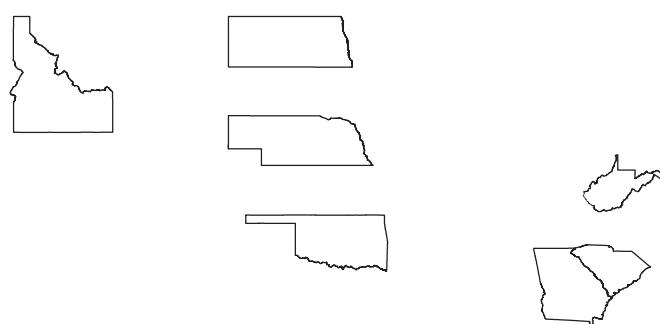
```
. keep if _merge == 3
(4 observations deleted)

. drop _merge
```

We can now create a map using `spmap`. By default, `spmap` will map the boundaries of all the states in the data set because they are listed in the `id` variable. We subset our data to only observations that are capital cities using the `select()` argument with `keep` and the `if` qualifier, leaving just one observation per id. Spatial points can easily be added to maps with the `points()` argument, using the variables holding their longitude (`v6`) and latitude (`lat`) information as the inputs for the `x()` and `y()` coordinates, respectively. Before mapping, we create a new `poprel` variable that scales population by a million. Each capital is represented by a solid circle and its size is proportional to its population (`poprel`), which we specify in the `proportion()` argument. We include `psize(absolute)` so that the markers represent absolute numbers, and are not defined in relation to the minimum and maximum values of `poprel`. We similarly normalize the range of values for `poprel` using `prange` to create a scale from 0 to 8 (8 approximates the maximum value of `poprel`, 8.2) so all states make an equal contribution. As with other graphs, we add a title to the graph using the `title()` option.

```
. generate poprel = pop / 1000000
. spmap using uscoord if capital == 2, id(id) point(select(keep if capital == 2) ///
> by(capital) x(v6) y(lat) proportional(poprel) prange(0 8) ///
> psize(absolute)) title("US state capitals") gsize(2.6)
```

US state capitals



As another example, we create a map of California's seven most populated cities. To extract these cities from the data, we use the `gsort` command to place the population variable in descending order. We then assign observations within each state an order number using Stata's built-in indexing variable `_n` and store this number in variable `citysize`. Because the `spmap` command only accepts unique identifiers (i.e., the `id` can only appear once), we also set the `id` of noncapital cities to missing for this exercise.

To limit the boundaries of our map to those of California, we add `if STUSPS=="CA"` to the first line of our `spmap` command. We further specify within the `select()` suboption under `label()` that only the top seven most populated cities in California should be labeled. We also include this qualifier in the `point` option. If we do not add this qualifier, Stata will add labels and points for all cities in the data set to our map. We tailor the size and placement of the labels, using `gap` (gap between point and label), `size` (text size of label), and `position` (clockwise position of label in relation to point), while also increasing the character length to ensure none of the city names are cut off. The suboptions `x()` and `y()` specify longitude and latitude, instructing Stata where to place both the labels and points on the map.



```

. gsort STUSPS -pop
. by STUSPS: generate citysize = _n
. replace id = . if capital != 2
(953 real changes made, 953 to missing)
.

```

```
. spmap using uscoord if STUSPS == "CA", id(id) ///
>           label(select(keep if citysize <= 7 & STUSPS == "CA")) ///
>           x(v6) y(lat) label(name) gap(*3 ..) ///
>           size(*1.25 ..) position(2) length(21)) ///
>           point(select(keep if citysize <= 7 & STUSPS == "CA") ///
>           by(citysize) x(v6) y(lat)) ///
>           title("Largest cities of California") gsize(3.5)
```

It is instructive to consider what the spatial polygon data look like in Stata. To do this, we open the `uscoord.dta` file that we have been using for the coordinates of each state. We can check the number of coordinates used to create the map of US states by looking at the total number of observations with nonmissing data on both the *x* and *y* coordinates. We display the first five of these coordinates.

```
. use uscoord, clear
. count if _X ~= . & _Y ~= .
294,518
. list in 1/5
```

	<u>_ID</u>	<u>_X</u>	<u>_Y</u>
1.	1	.	.
2.	1	-82.643198	38.16909
3.	1	-82.642997	38.16956
4.	1	-82.639054	38.171114
5.	1	-82.625457	38.170491

We observe that the map of US states consists of 294,518 pairs of coordinates. The `spmap` command connects these points to construct maps.

Spatial data contain information about patterns over space and can be visualized through maps. While **spatial point data** represent the locations of events as points on a map, **spatial polygon data** represent geographical areas by connecting points on a map.

### 7.2.3 UNITED STATES PRESIDENTIAL ELECTIONS

We now color a map of the United States using the 2008 presidential election results. The election data were introduced in chapter 4. The names and descriptions of variables in the data file `pres08.dta` are given in table 4.1.

We will color each state in two ways. First, we use blue for the states won by Obama and red for the states won by McCain. This will produce a map just like figure 4.1 with “blue and red

states.” Second, we compute the two-party vote share and color our map using a gradient that corresponds to the Democratic vote share. In this way, the color of a state reflects the degree of support for Obama versus McCain. The following code loads the data set and cleans the data for merger. We also create a binary variable to indicate a Democratic victory (`demwin`) and compute the vote share for the Democratic party (`demvote`). We then merge the election data into our spatial data set, keeping only matched cases.

```
. use pres08, clear
. rename statename NAME
. * binary win and vote share
. generate demwin = obama > mccain
. generate demvote = obama / (obama + mccain)
. merge 1:1 NAME using usdb
(note: variable NAME was str20, now str44 to accommodate using data's values)

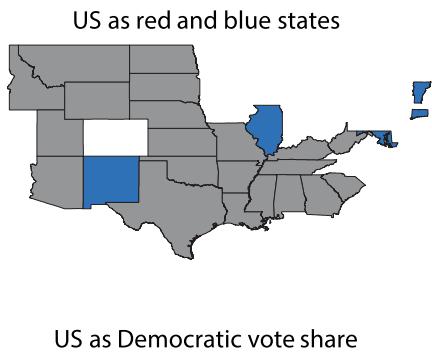
      Result          # of obs.

not matched           2
      from master       2 (_merge==1)
      from using         0 (_merge==2)
matched                49 (_merge==3)

. keep if _merge == 3
(2 observations deleted)
```

For our “blue and red states” map, we use our binary variable together with a dichotomized color scheme, where the states Obama won appear blue and those won by McCain are shown as red. The `spmap` command sorts the data to be plotted in ascending order. Thus, it plots the states where McCain won first (`demwin` equals 0), followed by the states where Obama won. Consequently, we first specify red, followed by blue in the `fcolor()` argument. We also draw a basic choropleth map, where we link the intensity of color to the Democratic share of the vote. By default, `spmap` will divide the data into four groups (or quartiles), on which the map fill will be based. The number of groups can be increased using the `clnumber()` option.

```
. * US as red and blue states
. spmap demwin using uscoord, id(id) fcolor(red blue) ///
>           title("US as red and blue states") legend(off) name(usblue, replace)
. * US in gradients of blue
. spmap demvote using uscoord, id(id) clnumber(8) fcolor(Blues) ///
>           title("US as Democratic vote share") legend(off) ///
>           name(demshare, replace)
. graph combine usblue demshare
```



The full-color versions of the maps are on page C6. Gray replaces red in this displayed version. The top map shows that Obama won many states on the West and East Coasts, whereas McCain was particularly strong in the Midwest. However, the bottom map illustrates that no state is completely dominated by either Democrats or Republicans. Each state has both types of voters and it is the winner-take-all electoral system that is responsible for characterizing each state as either a blue or a red state.

#### 7.2.4 EXPANSION OF WALMART

Shifting from politics to business, we next examine the expansion of Walmart, a successful American multinational chain of retail discount department and warehouse stores.<sup>4</sup> Walmart opened its first store in 1962 in Rogers, Arkansas. Over the next several decades, it opened many stores within the United States and then around the world. Walmart has become one of the largest retail multinational companies in the world. Table 7.3 shows the names and descriptions of variables in the Walmart store opening data, `walmart.dta`. This data set contains spatial and temporal information about Walmart store openings, from the first opening on March 1, 1962 until August 1, 2006.

We begin by plotting all of the store locations on a map. The data set contains three different types of stores, represented by the variable `type`. `Wal-MartStore` represents a standard Walmart store, whereas `SuperCenter` is a standard Walmart store as well as a full supermarket. Walmart Supercenters often include pharmacies, garden shops, car service centers, and other specialty centers. We also plot `DistributionCenter` data, representing stores that distribute food and goods to standard Walmart stores and Supercenters. In order to draw the boundaries for each state, we merge the `walmart` data set with the `usdb` spatial data.

<sup>4</sup>This section is in part based on Thomas J. Holmes (2011). “The diffusion of Wal-Mart and economies of density.” *Econometrica*, vol. 79, no. 1, pp. 253–302.

**Table 7.3.** Walmart Store Opening Data.

Variable	Description
opendate	opening date for the store
staddress	street address of the store
city	city of the store
state	state of the store
type	store type ( Wal-MartStore, SuperCenter, DistributionCenter)
_X	longitude of the store
_Y	latitude of the store

The data set contains spatial and temporal information about Walmart store openings from the first opening on March 1, 1962 until August 1, 2006.

```
. use walmart, clear
. merge 1:1 _n using usdb
      Result          # of obs.

not matched           3,202
      from master       3,202  (_merge==1)
      from using          0  (_merge==2)
matched                49  (_merge==3)
```

One may notice that the `merge` command is unlike others we have previously used because the shared identifier on which the merge is based is the observation number (`_n`). This means that the first row in the master data set will be matched with the first row of the using data set. The two data sets share no other variables and it is not necessary for the columns to be linked by another identifier, such as `state`, because our plots only require the variable names for generating points and boundaries. In fact, if one tries to do a many-to-one merge based on `state`, Stata will generate an error when using `spmap` because it expects `id` (our `state` identifier in the `usdb` data) to be unique (i.e., only one observation for each value).

While the `id` variable in the `usdb` data set is used to draw the boundaries of each state, each Walmart store is a point within the United States, given by the latitude and longitude coordinates in the Walmart data. We use these coordinates in the `spmap` command to specify the location of each store with the `point()` argument. To distinguish the three types of stores, we use different colors—red for standard Walmart stores, green for Supercenters, and blue for Distribution Centers. We lighten the color of the circles representing different stores for clarity when circles overlap. As before, the `proportional` argument can be used to specify the size of the points. We create a separate variable called `propsize` so Distribution Centers, which are fewer than the other two types, will be represented by larger circles than

stand out. We add a legend using `legenda(on)` and slightly reduce the size of the displayed graph using the `gsize` option. See page C6 for the full-color version.

```
. generate propsize = cond(type == 3, 3, .5)
. spmap using uscoord, id(id) point(by(type) ///
>           x(_X) y(_Y) fcolor(green*.8 blue*.5 red*.8) ///
>           size(small) proportional(propsize) legenda(on)) ///
>           legend(size(medlarge)) gsize(2.6)
```

Walmart store  
Super Center  
Distribution Center

The map clearly shows the business strategy of Walmart. While Supercenters are widespread throughout the Midwest and South, they appear less prevalent in the Northeast and the West Coast, as well as in urban centers more generally. In these areas, Walmart has chosen not to expand past the standard discount store format.

### 7.2.5 ANIMATION IN STATA

The previous analysis of Walmart store openings ignored the temporal dimension even though the data set contains the opening date. By examining the spatial-temporal patterns rather than spatial patterns alone, we can better understand how Walmart expanded its stores over time. What visualization strategy should we employ to achieve this goal? We can create a series of maps over time, showing all stores at various points in time.

To do this, we can use a loop to subset the data by specified dates and then produce a map of Walmart stores like the one shown above. We just need to place our previous code within the loop. Before we start the loop, we create a year variable, indicating the year the Walmart store opened. Because the `opendate` variable is a string, we convert it to a date variable using the `date()` function first introduced in section 4.1.3. We specify "YMD" because `opendate` lists the year, followed by the month and day. In the `spmap` command we use the `select()` suboption under `point()` to restrict the points to stores that opened on or

before the specified date and then plot their locations on a map. We create a map for every ten years (see also page C7).

```
. generate openyear = year(date(opendate, "YMD"))

. forvalues i = 1975(10)2005 {
    2.      spmap using uscoord, id(id) point(select(keep if openyear <= `i') ///
>           by(type) x(_X) y(_Y) fcolor(green*.8 blue*.5 red*.8) ///
>           size(small) proportional(propsize)) ///
>           legend(size(medlarge) title("`i'") name(yr`i', replace)
    3. }
. graph combine yr1975 yr1985 yr1995 yr2005
```

1975

1985

1995

2005

A better method to visualize spatial-temporal data is *animation*, which dynamically shows how geographical patterns change over time. To create animated graph files, Stata relies on external programs. There are separate options for Unix and Windows.

In either case, the first thing we have to do is create the series of maps we would like to animate. For this example, we want a map showing the number of Walmart stores open in each year so we loop over the values of our `openyear` variable. We store these values in local `y` by using the `levelsof` command. We also create a local called `count`, which will simply number each map sequentially with each loop iteration. We again subset observations to stores that opened in that year or in the years prior. Before running our loop, we use the `mkdir` command to create a new directory to hold the graphs and minimize clutter in our current directory. Each graph is converted to a .png file using the `graph export` command.

```

mkdir graphs
levelsof openyear, local(y)
local count = 1
foreach i of local y {
    spmap using uscoord, id(id) point(select(keep if openyear <= `i') ///
    by(type) x(_X) y(_Y) fcolor(green*.8 blue*.5 red*.8) ///
    size(small) proportional(propsize) legenda(on)) ///
    legend(size(medlarge)) title("`i'")

    graph export graphs/wm`count'.png, replace
    local count = `count' + 1
}

```

We will use the free program `ffmpeg`, which can be downloaded at <https://ffmpeg.org/>. The package is available for major platforms, including Windows and Mac OS (see, e.g., [https://github.com/adaptlearning/adapt\\_authoring/wiki/Installing-FFmpeg](https://github.com/adaptlearning/adapt_authoring/wiki/Installing-FFmpeg)). For Windows, programs can be called from within Stata with the `winexec` command and the exact file location. We then pass different options to the `ffmpeg` program in the command line. Both `winexec` commands should be entered as a single line.

```

winexec "C:\Program Files\ffmpeg\bin\ffmpeg.exe" -i graphs/wm%d.png -b:v 512k
        graphs/wm.mpg
sleep 10000
winexec "C:\Program Files\ffmpeg\bin\ffmpeg.exe" -r 1 -i graphs/wm.mpg -t 43 -r 1
        graphs/wm.gif

```

The first `winexec` command creates a `.mpg` file out of the graphs. The second `winexec` command converts the `.mpg` file to an animated `.gif` file. We add a `sleep` command to tell Stata to wait for 10,000 milliseconds (10 seconds) before running the second `winexec` command to allow the first `winexec` command to finish. In both `winexec` commands, the “`-i`” option specifies the input while `%d` says to match any pattern of digits. The “`-b:v`” option sets the buffer rate and the “`-r`” option sets the frames per second. Finally, “`-t`” is the total duration. We can add “`-y`” to overwrite existing files.

Mac users may need to install a package manager like Homebrew before they are able to install `ffmpeg` (see <https://brew.sh/>).<sup>5</sup> Once `ffmpeg` has been successfully downloaded and installed, Terminal can be called through Stata using the `shell` command. It is again important that the directory where the maps are saved is currently set as the working directory, or the pathway must be included in the command line as was done in our `winexec` command. After running the `shell` commands below, users will find the respective `.mpg` and `.gif` files in the specified directory.

<sup>5</sup>Users can install `ffmpeg` by entering `brew install ffmpeg` in Terminal. Typing `brew update && brew upgrade ffmpeg` will then install any updates.

```
shell /usr/local/bin/ffmpeg -i graphs/wm%d.png -b:v 512k graphs/wm.mpg
sleep 10000
shell /usr/local/bin/ffmpeg -r 1 -i graphs/wm.mpg -t 43 -r 1 -y graphs/wm.gif
```

In a Unix environment, the process does not require additional programs.<sup>6</sup> Unix has a `convert` command that will convert image files to a different type of image file. We could use this command to create a `.gif` file out of a list of Stata graph files.

```
* get list of graphs using Unix
local input: dir "" files "*.png"
shell convert -delay 150 -loop 0 `macval(input)' wm.gif
```

We can open the `.gif` file by going to the directory and clicking on the file. While watching, we see quite clearly the Southern origins of the Walmart franchise and its gradual spread throughout the region in the 1970s and 1980s. Particularly striking is the speed of the mid-1990s expansion throughout the rest of the country, as well as when and where new Distribution Centers are established in anticipation of regional expansion.

### 7.3 Textual Data

The widespread use of the Internet has led to an astronomical amount of digitized textual data accumulating every second through email, websites, and social media outlets. The analysis of blog sites and social media posts can give new insights into human behavior and opinions. At the same time, large-scale efforts to digitize published articles, books, and government documents have been underway, providing exciting opportunities to revisit previously studied questions by analyzing new data.

#### 7.3.1 THE DISPUTED AUTHORSHIP OF *THE FEDERALIST PAPERS*

While new opportunities for text analysis have grown in recent years, we begin by revisiting one of the earliest examples of text analysis in the statistics literature. We analyze the text of *The Federalist*, more commonly known as *The Federalist Papers*.<sup>7</sup> *The Federalist*, whose title page is displayed in figure 7.5, consists of 85 essays attributed to Alexander Hamilton, John Jay, and James Madison from 1787 to 1788 in order to encourage people in New York to ratify the newly drafted US Constitution. Because both Hamilton and Madison helped draft the Constitution, scholars regard *The Federalist Papers* as a primary document reflecting the intentions of the authors of the Constitution.

*The Federalist Papers* were originally published in various New York state newspapers under the pseudonym of “Publius.” For this reason, the authorship of each paper has been

<sup>6</sup>While Windows now has a bash program that will run Unix scripts, it is difficult to use it with Stata, partially because the directory structure is different in the two programs.

<sup>7</sup>This section is in part based on F. Mosteller and D.L. Wallace (1963). “Inference in an authorship problem.” *Journal of the American Statistical Association*, vol. 58, no. 302, pp. 275–309.

Figure 7.5. The Title Page of *The Federalist* Vol. 1.  
Source: Library of Congress.

the subject of scholarly research. According to the Library of Congress,<sup>8</sup> experts believe that Hamilton wrote 51 essays while Madison authored 15.<sup>9</sup> In addition, Hamilton and Madison jointly authored 3 papers whereas John Jay wrote 5.<sup>10</sup> The remaining 11 essays were written by either Hamilton or Madison, though scholars dispute which one.<sup>11</sup> In this section, we analyze the text of *The Federalist Papers* to predict their authorship.

The text from each of the 85 essays has been scraped from the Library of Congress website and stored in a `fpXX.txt` file, where XX represents the essay number ranging from 01 to 85. *Scraping* refers to an automated method of data collection from websites using a computer program. Each data file contains the textual data of its corresponding essay. See table 7.4, which displays the first and last sentences of *The Federalist Paper* No. 1 as an example.

Before analyzing the data, we need to load it into Stata. We will initially load the data so we have three variables for each essay: the full text of the essay (`txt`), the original `fpXX.txt` file name (`file`), and the essay number (`fedno`). Our preliminary data set will therefore have three columns (variables) and 85 rows (one for each essay).

To build our data set, we create an empty temporary file call `fedpapers`. We then use the `fs` package and its `fs` command to identify all `fpXX.txt` files within our `federalist` subdirectory. The `fs` command makes a list of all files in the current folder, so it is important that the working directory is correct. The correct pathway can also be included as part of the `fs` command (e.g., `fs federalist/fp*.txt`). We replace the XX in our file name with

<sup>8</sup>See the website <https://guides.loc.gov/federalist-papers/full-text>. (<https://guides.loc.gov/federalist-papers/full-text>)

<sup>9</sup>*The Federalist Papers* known to be written by Hamilton: Nos. 1, 6–9, 11–13, 15–17, 21–36, 59–61, and 65–85. Papers known to be written by Madison: Nos. 10, 14, 37–48, and 58.

<sup>10</sup>*The Federalist Papers* known to be jointly written by Hamilton and Madison: Nos. 18–20. *The Federalist Papers* known to be written by John Jay: Nos. 2–5 and 64.

<sup>11</sup>*The Federalist Papers* with disputed authorship are Nos. 49, 50–57, 62, and 63.

**Table 7.4.** *The Federalist Papers Data.*


---

AFTER an unequivocal experience of the inefficiency of the subsisting federal government, you are called upon to deliberate on a new Constitution for the United States of America.

⋮

This shall accordingly constitute the subject of my next address.

---

The data consist of the raw text of each of the 85 essays in *The Federalist*. The first and last sentences of *The Federalist Paper* No. 1 appear here as an example.

the wildcard character (\*) so all files starting with "fp" and ending with ".txt" are identified. The **fs** package offers an efficient way to identify not only all files within a folder, but it can also be used when working across multiple folders, using the **folders** command. The full list of file names is returned in **r(files)**, which we plug into our loop, where we append the text from each of the .txt files.

```
clear
* change directory and create temporary file
cd federalist
 tempfile fedpapers
 save `fedpapers', emptyok

* loop through all essays
fs fp*.txt
foreach f in `r(files)' {
    clear
    set obs 1
    generate file = "`f'"
    generate strL txt = fileread("`f'")
    generate fedno = substr(file, 3, 2)
        destring fedno, replace
    append using `fedpapers'
    save `"`fedpapers`"', replace
}
use `fedpapers', clear
```

In our code, we loop through each essay by first creating a data set with one observation (**set obs 1**). This ensures that all text from the .txt file will be stored in just one cell. We then use the **fileread** function with **generate** to store the full text of the file in variable **txt**.

We specify that this variable is a long string using `strL` in our `generate` command. We extract the essay number from the file name, requesting a string value that starts with the third character of `fed` and has a length of two, converting this to a numeric value using the `destring` command. Finally, the data are appended to our `fedpapers` data set. We save the updated data before moving onto the next essay.

For our text analysis, we use two user-written packages. These are **txttool** and **wordcloud**. They can be installed by running `net search txttool` and `net search wordcloud`. We begin by preprocessing our corpus (or body of text). By default, the `txttool` command will convert all text to lowercase and will keep only letters, numbers, and whitespace. All other characters will be removed. We tell Stata to replace our existing variable (`txt`) rather than generating a new one. We create a copy of the `txt` variable called `fed`, on which we will conduct further processing, while saving `txt` for later analysis.

```
. * make lowercase; keep letters, numbers, whitespace
. txttool txt, replace
Input: 14957 unique words, 191225 total words
Output: 8667 unique words, 187228 total words
Total time: 3.394 seconds

. generate fed = txt
```

Using `fed`, we remove commonly used words (or stop words), such as `a` and `the`. The `txttool` package comes with its own list of stop words in a file called `stopwordexample.txt`. The beginning of the list includes: `a`, `about`, `above`, `after`, `again`, `against`, `all`, `am`, and `an`. We pass the list through the `stopwords()` option. Also, we stem each word using the `stem` option. Stemming words strips away prefixes and suffixes to produce stem words so that different forms of the same word can be recognized. For example, the stem form of “government” is “govern.”

```
. * remove stopwords and stem words
. txttool fed, stopwords("stopwordexample.txt") stem replace
Input: 8667 unique words, 187228 total words
Output: 5010 unique words, 86064 total words
Total time: 8.101 seconds
```

As `txttool` does not remove numbers from its list of words, we do this ourselves. Starting in Stata 14, Stata supports unicode regular expressions, enabling us to use the `replace` function called `ustrregrexra()`. Within the parentheses, we specify our string variable (`fed`), followed by the pattern we want to replace in that string (here, any number), and then what should replace that pattern when found. In this case, we use double quotations so numbers are removed without a replacement. Using square brackets with “`0-9`” inside specifies that every number should be replaced. The plus sign after the square brackets looks for numbers with multiple digits.

```
. * remove numbers
. replace fed = ustrregexra(fed, "[0-9]+", "")
(48 real changes made)
```

Now that we have removed numbers and stop words from our stemmed text, we save our data. We can extract the processed text of a specific essay stored in `fed` by referring to the essay number in variable `fedno`.

```
save fedpapers, replace
```

```
. * truncated output of essay no. 10
. list fed if fedno == 10
```

fed
56. among numer advantag promis wellconstruct union none deserv accur develop tendenc break c..

Compare this preprocessed document with the corresponding section of the original text that follows.

```
AMONG the numerous advantages promised by a well-constructed Union, none
deserves to be more accurately developed than its tendency to break and
control the violence of faction. The friend of popular governments never
...
...
```

We observe that all preprocessing was done as specified in our prior code. That is, all letters were transformed to lowercase, punctuation marks such as hyphens and commas were taken out, stop words were removed, and words were reduced to their stem form (e.g.,numerous to numer and promised to promis).

### 7.3.2 TOPIC DISCOVERY

One quick way to explore textual data is to simply count occurrences of each word or term. The number of times a particular word appears in a given document is called *term frequency* (*tf*). The frequency for words in our text is easily calculated using the `wordfreq` command, which is included in the `wordcloud` package. This command requires only our text variable so we preserve our data and keep variable `fed`. We have to save the truncated data before we can include it in our `wordfreq` command. Placing the words in descending order using the newly created variable `freq` and the `gsort` command, we list the top 10 words used across all documents as a whole.

```
. preserve
.
    keep fed
.
    save freq, replace
.
    wordfreq using freq.dta
.
    gsort -freq
.
    list word in 1 / 10
```

	word
1.	state
2.	will
3.	govern
4.	power
5.	mai
6.	constitut
7.	nation
8.	on
9.	peopl
10.	can

```
.
    erase freq.dta
.
    restore
```

By using a loop, we can also create a data set that appends the term frequency for each individual document, rather than computing the frequency across all documents. This requires only slight modification of our previous code. We then clean the data returned from the `wordfreq` command by removing any numbers or special characters using the `ustrregrexra()` function. Within this function, we tell Stata to keep only lower or uppercase letters. We also drop any words that are empty strings or have fewer than three characters.

```
. clear
.
tempfile termfreq
.
save `termfreq', emptyok
.
.
* loop over all 85 papers for word frequency
.
forvalues i = 1 / 85 {
.
    use fedpapers, clear
.
    * keep one essay and compute frequency
```

```

.
    keep if fedno == `i'
.
    keep fed
.
    save freq, replace
.
    wordfreq using freq.dta
.
    * create paper number variable
.
    generate fedno = `i'
.
    * append with other essays
.
    append using `termfreq'
.
    save `termfreq', replace
.
}
.
use `termfreq', clear
.
.
* data cleaning
.
replace word = ustrregexra(word, `"[^a-zA-Z]"', "")
.
drop if word == ""
.
drop if length(word) < 3

```

As an example, we can see the 10 most frequently used words in Paper No. 1.

```

.
gsort fedno -freq
.
list word in 1 / 10 if fedno == 1

```

	word
1.	will
2.	mai
3.	constitut
4.	govern
5.	interest
6.	men
7.	shall
8.	union
9.	state
10.	good

Our analysis of word frequency critically relies on the commonly used *bag-of-words* assumption, which ignores the grammar and ordering of words. This means that our analysis is unlikely to detect subtle meanings of texts. The distribution of *term frequency* (tf) should, however, allow us to infer *topics* discussed in the documents. A common way to visualize this distribution is a *word cloud* where more frequently used words appear in a larger font. Word clouds can provide a useful visualization tool for quickly extracting key words.

Like clustering, covered in section 3.7, topic discovery is an example of *unsupervised learning* because we lack access to true information about topic assignment. That is, we do not know, *a priori*, what topics exist in the corpus and characterize each document. We wish to discover topics by analyzing the distribution of term frequency within a given document and across documents. In contrast, in *supervised learning*, researchers use a sample with an observed outcome variable to learn about the relationship between the outcome and predictors. For example, we may have human coders read some documents and assign topics. We can then use this information to predict the topics of other documents that have not been read. Clearly, the lack of information about outcome variables makes unsupervised learning problems more challenging than supervised problems.

We will draw some simple word clouds using Stata. Our code is an adjustment to that provided by the `wordcloud` command. To generate a word cloud using this command, users simply need to include the word and frequency variables (`wordcloud word freq`). The package creators include a `showcommand` option that displays the full set of commands used to generate the word cloud. They welcome users to modify the code as needed. Users are encouraged to see more at `help wordcloud`. Because we focus on a limited set of words in our example, it was necessary to adjust the code that sets the word position and size. Although some users may not find adjustments necessary for larger bodies of text, we walk readers through our tailored code.

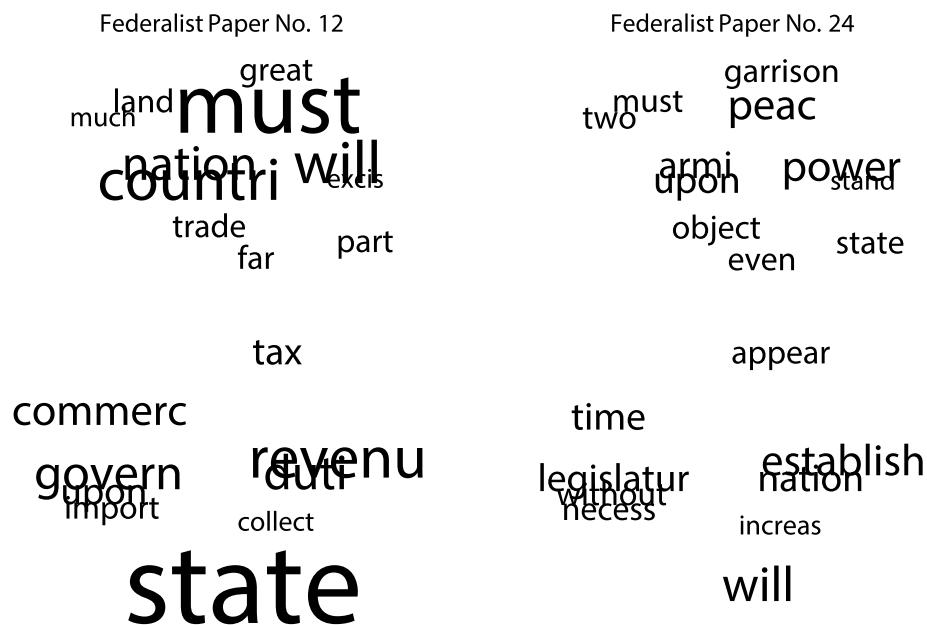
We begin by visualizing *The Federalist Papers* Nos. 12 and 24 with word clouds in order to infer their topics. Both papers are known to have been authored by Alexander Hamilton. We create a loop to go through each paper. The `gsort -freq` command first places the words in descending order by frequency. For readability, we limit the maximum number of words to be plotted to the 20 most frequently used words in a document (`keep in 1 / 20`). The placement of words in a word cloud is random so we use `runiformint` to generate random *x*- and *y*-coordinates for our scatterplots. To minimize the possibility that words will overlap with one another, we set the values of the random numbers to range from 1 to 40. We use `levelsof` to get the unique set of values for the frequency variable so we can plot a separate `scatter` command for each frequency value. Rather than typing each `scatter` command separately, we create a local macro that holds the value of each `scatter` command. Lastly, we make some modifications to the graph itself by removing the *x*- and *y*-lines and labels. We also expand the margins to allow more room for the words. We do not show the full output in the interest of space.

```
. * federalist paper Nos. 12 and 24
. local papers "12 24"
. foreach paper of local papers {
.   preserve
.     keep if fedno == `paper'
.     gsort -freq
.     keep in 1 / 20
.     set seed 12345
.     generate xunif = runiformint(1,40)
.     generate yunif = runiformint(1,40)
.     levelsof freq, local(f)
```

```

.
local s ""
foreach n of local f {
    local s `"$s' (scatter yunif xunif if freq == `n',
> msymbol(i) mlabpos(0) mlab(word)
> mlabsize(`n'))"
}
twoway `s', xscale(noline) yscale(noline) ///
xtitle("") ytitle("") xlabel(none) ylabel(none) ///
graphregion(margin(l+15 r+15 t+5 b+10) color(white)) ///
title("Federalist Paper No. `paper'", size(huge)) ///
legend(off) name(fed`paper', replace)
restore
}
graph combine fed12 fed24

```



The comparison of the two word clouds shows that the left plot for Paper No. 12 contains words related to the economy such as *revenu* (the root form of *revenue*), *commerc* (*commerce*), *trade*, *import*, *tax*, and so on. In contrast, the right plot for Paper No. 24 contains more words about security, including *power*, *peac* (the root form of *peace*), and *garrison*.

These discovered topics are indeed consistent with the actual contents of the papers. Paper No. 12 is entitled “The Utility of the Union in Respect to Revenue” and discusses the economic benefits of forming the union. In contrast, the title of No. 24 is “The Powers Necessary to the Common Defense Further Considered” and it discusses the creation of a national army as well as the relationship between legislative power and national forces.

In this analysis, we visualized the distribution of term frequency within each document. However, a certain term’s high frequency within a document means little if that term often

appears across the documents of the corpus. To address this issue, we should downweight the terms that occur frequently across documents. This can be done by computing the statistic called *term frequency-inverse document frequency*, or *tf-idf* in short. The tf-idf is another measure of the importance of each term in a given document. For a given document  $d$  and term  $w$ , we define  $\text{tf-idf}(w, d)$  as follows:

$$\text{tf-idf}(w, d) = \text{tf}(w, d) \times \text{idf}(w). \quad (7.2)$$

In equation (7.2),  $\text{tf}(w, d)$  represents *term frequency* or the number of occurrences of term  $w$  in document  $d$ . In some cases, we convert  $\text{tf}(w, d)$  to the log scale when it takes a positive value. Note that  $\text{tf}(w, d)$  equals zero when term  $w$  never occurs in document  $d$ .

The other factor in equation (7.2),  $\text{idf}(w)$ , is the *inverse document frequency* or *idf*, which is typically defined as

$$\text{idf}(w) = \log \frac{N}{\text{df}(w)}.$$

In this equation,  $N$  is the total number of documents and  $\text{df}(w)$  is the *document frequency* or the number of documents that contain term  $w$ . This implies that  $\text{idf}(w)$  takes a smaller value when term  $w$  is used more frequently across documents. As a consequence, common terms across documents receive less weight in tf-idf.

We can compute the tf-idf measure by calculating each of its component parts. We already know the number of documents, and the `freq` variable gives us the term frequency of each word in a document. The document frequency is calculated by counting the number of documents in which each word appears. If we want to weight the term frequency, we can calculate the total number of words in each document. We do this by summing `freq` within each document. Finally, we compute both the unweighted and weighted versions of the tf-idf. In the weighted version, term frequency  $\text{tf}(w, d)$  will be divided by the total number of terms in document  $d$ .<sup>12</sup>

```
. sort word fedno
. by word: egen df = count(fedno)
. bysort fedno: egen tf = sum(freq)
. generate tfidfw = freq * (ln(85 / df) / ln(2))
. generate tfidfw = (freq / tf) * (ln(85 / df) / ln(2))
```

We list the 10 most important terms for *The Federalist Papers* No. 12 and No. 24 using the weighted tf-idf measure. We again use `gsort` to sort the data in descending order of tf-idf values. And we use the weighted tf-idf measure for the remainder of the chapter.

```
. gsort fedno -tfidfw
. by fedno : generate tf_rank = _n
```

<sup>12</sup>Often, including in R, the log in base 2 is used with the inverse document frequency. We do the same by taking the log and then dividing by the log of 2.

```
. list word tfidfw if fedno == 12 & tf_rank <= 10
```

	word	tfidfw
6056.	revenu	.0184961
6057.	patrol	.0183125
6058.	contraband	.0183125
6059.	excis	.0182116
6060.	coast	.0154554
<hr/>		
6061.	trade	.0143
6062.	per	.0137841
6063.	tax	.0125722
6064.	gallon	.0122084
6065.	cent	.0122084

```
. list word tfidfw if fedno == 24 & tf_rank <= 10
```

	word	tfidfw
13236.	garrison	.0283523
13237.	settlement	.0187609
13238.	dockyard	.0187609
13239.	spain	.015766
13240.	armi	.0147642
<hr/>		
13241.	frontier	.0141762
13242.	arsen	.0125073
13243.	western	.0124926
13244.	post	.0118245
13245.	nearer	.0111547

The results show that the most important terms for *The Federalist Paper* No. 12 concern the economy whereas those for Paper No. 24 relate to security policies, though such word association is done by the researcher.

The analysis of documents based on term frequency relies on the **bag-of-words** assumption that ignores the order of words. To measure the relative importance of a term in a document, we can compute the **term frequency-inverse document frequency** (tf-idf), which represents the relative frequency of the term inversely weighted by the number of documents in which the term appears (document frequency).

### 7.3.3 DOCUMENT-TERM MATRIX AND CLUSTERS

We next cluster all the essays written by Hamilton based on the tf-idf measure. Following section 3.7, we apply the  $k$ -means algorithm. We have to reshape the data so that the words become variables that hold their tf-idf values. This transformation corresponds to a *document-term matrix*, which is a rectangular array with rows representing documents (or papers, in this case) and columns representing unique terms. In other words, there is a column for every word. The  $(i, j)$  element of this matrix gives the tf-idf value of the  $j$ th term (column) in the  $i$ th document (row). We can also flip rows and columns and convert a document-term matrix into a *term-document matrix*, where rows and columns represent terms and documents, respectively.

In the next section, we will see that we can transform the textual data into a document-term matrix using the `bagw` option within the `txttool` command. However, it is important to introduce Stata's broader `reshape` command that extends beyond text analysis. This command allows us to transpose data sets, where rows become columns and columns become rows. These transformations restructure data from what is called long to wide format, and wide to long format, respectively. Because data are not always in the format required for analysis, users are encouraged to consult `help reshape`. For example, when using the time-series operators introduced in section 4.1.3, data must be in long format, where variables are stored in columns and observations of time (e.g., year or day) are held in rows. Knowing the structure of data is indispensable for any analysis. We present only a brief application of the `reshape` command, converting words previously stored in rows into columns.

To be sure, the document-term matrix is not stored efficiently in Stata. Most words do not appear in all documents, so the matrix is typically *sparse*. That is, a vast majority of its entries are zero or missing because most terms appear only in a small number of documents. But even more, users are constrained by Stata's limitations on the number of variables. Because the document-term matrix holds each word as a separate variable, larger amounts of text can quickly exceed the current limits. Stata/IC has a limit of 2,048 variables, while Stata 14's MP and SE versions are both limited to 32,767 variables. Consequently, some users may be unable to transform the current data set into a document-term matrix, which holds more than 4,000 variables. The default setting of 5,000 variables in Stata MP and SE can be increased by using the `set maxvar` command.

For our analysis, we keep only the Hamilton papers before reshaping the data and running the  $k$ -means algorithm. Once the data are reshaped, we have to replace missing tf-idf values, which reflect sparsity, with 0's or Stata will warn us that there are insufficient observations to conduct our cluster analysis.

```

. generate hamilton = fedno==1 | inrange(fedno, 6, 9) | ///
>           inrange(fedno, 11, 13) | inrange(fedno, 15, 17) | ///
>           inrange(fedno, 21, 36) | inrange(fedno, 59, 61) | ///
>           inrange(fedno, 65, 85)
. keep if hamilton == 1
. keep fedno word tfidfw
. reshape wide tfidfw, i(fedno) j(word) string

```

```

. * replace missing values with zero
. foreach var of varlist tfidfw* {
.     replace `var' = 0 if `var' == .
. }
```

To run our *k*-means algorithm, we choose the number of clusters to be four, after some experimentation. While arbitrary, this choice produces clusters that seem reasonable. We set the number of iterations to 5. We start by grouping the documents randomly into four centers (`krandom`) and tell Stata to keep the centers for each variable (`keepcenters`). This will add four rows (observations) to the end of our data set that hold the centers for each variable across each cluster. We keep only these four observations and assign a cluster number using `_n`.

```

. cluster kmeans tfidfw*, k(4) start(krandom(12345)) keepcenters iter(5)
. keep if _clus == .
. generate cluster = _n
```

We next summarize the results by identifying the ten words with the highest tf–idf values in each cluster. We create a loop that first identifies the highest variable value in each row using the `rowmax` function with `egen`. We then create a variable that will hold the name of the word variable with that maximum value. After we identify the variable and store its name in `maxvar`, we replace the highest value with missing value `.a` so we can repeat the loop and find the second highest value, repeating this process to obtain the top 10 values. We perform minor data cleaning by removing the "tfidf" prefix from our variable names before displaying the results.

```

. local vars tfidfw*
. forvalues i = 1 / 10 {
.     * identify maximum row value and create variable to store name
.     egen double maxval`i' = rowmax(`vars')
.     generate maxvar`i' = ""
.     * identify which variable has the maximum value and store name
.     foreach var of varlist `vars' {
.         replace maxvar`i' = "`var'" if `var' == maxval`i'
.         replace `var' = .a if `var' == maxval`i'
.     }
.     * remove tfidf prefix
.     replace maxvar`i' = substr(maxvar`i', 7, .)
. }
```

```
. * combine variable names into one string
. egen topten = concat(maxvar*), p(" ", " )

. list topten
```

topten
1. court, senat, upon, presid, claus, armi, juri, offic, militia, appoint 2. revenu, taxat, debt, resourc, expens, loan, sum, pound, patrol, excis 3. vassal, feudal, baron, feudatori, nobl, diffus, sovereign, obedi, chiefli, neighborhood 4. guaranti, inequ, land, wealth, total, consumpt, popul, valu, valuat, quota

Examining the 10 most important terms at the centroid of each cluster suggests that cluster 2 relates to economic concerns, as indicated by terms like `revenu`, `taxat`, `debt`, `loan`, and so on. Cluster 3 addresses institutional design and cluster 1 appears to be concerned with judicial policies, including terms like `court`, `juri`, and `claus`. Comparing these topics with the actual contents of *The Federalist Papers* shows a fair degree of validity for the results of the *k*-means clustering algorithm.

We have been using *The Federalist Papers* to illustrate how text analyses can reveal topics. Of course, since we can easily read all of *The Federalist Papers*, the automated text analysis may not be necessary in this case. However, similar and more advanced techniques can be applied to a much larger corpus that humans would struggle to read in full over a short amount of time. In such situations, automated text analysis can play an essential role in helping researchers extract meaningful information from textual data.

### 7.3.4 AUTHORSHIP PREDICTION

As mentioned earlier, the authorship of some of *The Federalist Papers* is unknown. We will use the 66 essays attributed to either Hamilton or Madison to predict the authorship of the 11 disputed papers. Since each *Federalist* paper deals with a different topic, we focus on the usage of adjectives, adverbs, prepositions and conjunctions. In particular, we analyze the frequency of the following 10 words: `although`, `always`, `commonly`, `consequently`, `considerable`, `enough`, `there`, `upon`, `while`, and `whilst`. We select these words based on the analysis presented in the academic paper that inspired this section (see footnote 7). As a result, we will use the unstemmed text. We return to our original data set containing our merged documents, where all of the text from one document is stored in one cell. We then use the `bagw` option with the `txttool` command to create another document-term matrix. This time, however, the variable values will be the term frequency counts, not the tf-idf like in the previous section. Unless otherwise specified, `txttool` will add the prefix “`w_`” to our transformed terms. Before starting this task, users are reminded that they may have to increase the maximum number of variables using `set maxvar`.

```
. use fedpapers, clear
. keep fedno txt fed
```

```
. sort fedno
. txttool txt, bagw replace
No prefix chosen for bagged words; default prefix w_ used
Note that errors will result if new prefix/word combinations match any
existing variables
Input: 8667 unique words, 187228 total words
Output: 8667 unique words, 187228 total words
Total time: 500.381 seconds
```

To calculate the average term frequency separately for Hamilton and Madison across each author's entire body of documents, we first create the outcome variable by coding essays authored by Hamilton as 1 and those written by Madison as -1. We subset the data to keep only the 10 words we want to analyze, together with the authorship variables and the document numbers.

```
. generate hamilton = fedno == 1 | inrange(fedno, 6, 9) | ///
>      inrange(fedno, 11, 13) | inrange(fedno, 15, 17) | ///
>      inrange(fedno, 21, 36) | inrange(fedno, 59, 61) | ///
>      inrange(fedno, 65, 85)

. generate madison = fedno == 10 | fedno == 14 | inrange(fedno, 37, 48) | ///
>      fedno == 58

.

. generate author = 1 if hamilton == 1
(34 missing values generated)

. replace author = -1 if madison == 1
(15 real changes made)

. label define auth -1 "Madison" 1 "Hamilton"

. label val author auth

.

. keep txt-fed hamilton-author w_although w_always w_commonly ///
>      w_consequently w_considerable w_enough w_there w_upon w_while w_whilst
```

We count the words in the raw text variable to get the total number of words in each document. We then create term frequencies (per 1,000 words) for each word with the use of a local macro and loop, and view the results using tabstat.

```
. * calculate term frequencies for each word
. generate wdct = wordcount(txt)
```

.	local newlist "although always commonly consequently considerably enough there upon while whilst"	
.	foreach v of local newlist {	
2.	generate tfm_`v' = (w_`v' / wdct) * 1000	
3. }		
.	tabstat tfm*, by(author) col(stat) long nototal	
author	variable	mean
Madison	tfm_although	.2076971
	tfm_always	.1550597
	tfm_commonly	0
	tfm_conseq_y	.3470544
	tfm_consider	.124637
	tfm_enough	0
	tfm_there	.8569547
	tfm_upon	.1530825
	tfm_while	0
	tfm_whilst	.2944402
Hamilton	tfm_although	.0132307
	tfm_always	.563541
	tfm_commonly	.1986841
	tfm_conseq_y	.0192348
	tfm_consider	.4069352
	tfm_enough	.295359
	tfm_there	3.309133
	tfm_upon	3.296451
	tfm_while	.2747986
	tfm_whilst	.0051804

The results suggest that, relative to Madison, Hamilton prefers to use terms such as *there* and *upon*. In contrast, *consequently* and *whilst* are used often by Madison, but rarely appear in the essays authored by Hamilton. We will use the frequency of these four words as the predictors of a linear regression model in order to predict the disputed authorship of the 11 essays. We first fit the linear regression model introduced in section 4.2 to the 66 essays whose authorship is known. The resulting fitted model will predict the authorship of the 11 essays based on the four words’ frequencies.

To predict the authorship, we use the term frequency of the four words selected based on our preliminary analysis, i.e., *upon*, *there*, *consequently*, and *whilst*. These variables all have the prefix “tfm” in our data.

.	regress author tfm_upon tfm_there tfm_consequently tfm_whilst					
Source	SS	df	MS	Number of obs	=	66
Model	33.45546	4	8.363865	F(4, 61)	=	39.53
Residual	12.9081764	61	.211609449	Prob > F	=	0.0000
				R-squared	=	0.7216
				Adj R-squared	=	0.7033
Total	46.3636364	65	.713286713	Root MSE	=	.46001
<hr/>						
author	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
tfm_upon	.2239949	.0365989	6.12	0.000	.1508109	.2971789
tfm_there	.1276949	.0343861	3.71	0.000	.0589356	.1964541
tfm_consequently	-.5591844	.3133221	-1.78	0.079	-1.185711	.0673418
tfm_whilst	-.8388162	.4015358	-2.09	0.041	-1.641737	-.0358959
_cons	-.2723972	.1426954	-1.91	0.061	-.5577342	.0129398

The results are consistent with our prior analysis. The estimated coefficients for upon and there are positive while those for consequently and whilst are negative, implying that the first two words are associated with Hamilton whereas the latter are associated with Madison. Interestingly, the estimated coefficient for whilst has the largest magnitude. Holding the term frequency of the other three words constant, one additional use of whilst (per 1,000 words) in an essay decreases the predicted authorship score by .84. To put this number into perspective, we compute the standard deviation of fitted values by predicting the linear prediction and then summarizing the variable.

.	predict authpred, xb					
* subset to papers authored by Hamilton or Madison						
.	summarize authpred if inlist(author, -1, 1)					
Variable	Obs	Mean	Std. Dev.	Min	Max	
authpred	66	.5454545	.7174255	-1.677646	1.856135	
* standard deviation						
.	display r(sd)					
.	71742552					

We find that the magnitude of this coefficient is large and close to one standard deviation of fitted values. That is, one additional use of whilst (per 1,000 words) accounts for approximately one standard deviation of variation in our predicted value for the authorship score.

### 7.3.5 CROSS VALIDATION

How well does this model fit the data? We classify each essay using its fitted value and compute the *classification error*. For this, we create a new variable that equals 1 for a positive fitted value and  $-1$  for a negative fitted value. We then cross tabulate this variable with the author variable. The results represent the classification success rate (see section 4.1.3).

```
. generate d_authpred = cond(authpred < 0, -1, 1)
. tabulate author d_authpred
```

author	d_authpred		Total
	-1	1	
Madison	15	0	15
Hamilton	0	51	51
Total	15	51	66

The results show that the model perfectly classifies the authorship of these essays. Like the coefficient of determination introduced in chapter 4, however, this measure of prediction accuracy is based on *in-sample prediction*. That is, the same data we used to fit the model are again used for assessing the prediction accuracy. This is not necessarily a good idea because we can *overfit* a model to the data at hand. Overfitting occurs when a model captures idiosyncratic features of a specific sample while muddling up systematic patterns that exist across different samples.

Let us instead consider *out-of-sample prediction*. The idea is that we use new observations to assess the predictive performance of a model. In chapter 4, we performed out-of-sample prediction by forecasting election results using preelection polls. Similarly, here, we employ a procedure called *leave-one-out cross validation*. Specifically, we set aside one observation and predict its outcome variable value after fitting the model to the remaining observations. We repeat this procedure for each observation in the sample and compute the classification error. Cross validation enables us to assess the accuracy of model prediction without relying on in-sample prediction.

**Cross validation** is a methodology to assess the accuracy of model prediction without relying on in-sample prediction, which often leads to overfitting. Suppose that we have a sample of  $n$  observations. Then, the leave-one-out cross-validation procedure repeats the following steps for each observation  $i = 1, \dots, n$ :

1. Take out the  $i$ th observation and set it aside.
2. Fit the model using the remaining  $n - 1$  observations.
3. Using the fitted model, predict the outcome for the  $i$ th observation and compute the prediction error.

Finally, compute the average prediction error across  $n$  observations as a measure of prediction accuracy.

In Stata, we can cross validate using a *loop*, where each iteration fits the model to the data after excluding one observation, then predicts that observation's outcome variable value. A convenient way to set aside the *i*th observation is to use the `if` qualifier with the “not equal to” condition (`!=` or `~`). As we saw in section 4.3.4, the `predict` command can compute the predicted value  $\hat{Y}$  for sample observations. We replace the overall prediction variable with the prediction on the excluded observation. We store the papers authored by either Hamilton or Madison in a local macro called `papers`, which we then use to iterate our loop.

```
. generate excl_predict = .
. levelsof fedno if inlist(author, -1, 1), local(papers)
. foreach i of local papers {
.     regress author tfm_upon tfm_there tfm_consequently ///
>             tfm_whilst if fedno != `i'
.     predict auth_temp
.     replace excl_predict = auth_temp if fedno == `i'
.     drop auth_temp
. }
```

The results show that even when the cross validation procedure is used, the model continues to perfectly classify the authorship of each essay.

```
. generate excl_predict2 = cond(excl_predict > 0, 1, -1)
. tabulate author excl_predict2
```

author	excl_predict2		Total
	-1	1	
Madison	15	0	15
Hamilton	0	51	51
Total	15	51	66

We can now use this fitted model to predict the unknown authorship of the 11 essays. When we ran the `predict` command, Stata calculated the fitted value for the essays that are missing a value for `author`.

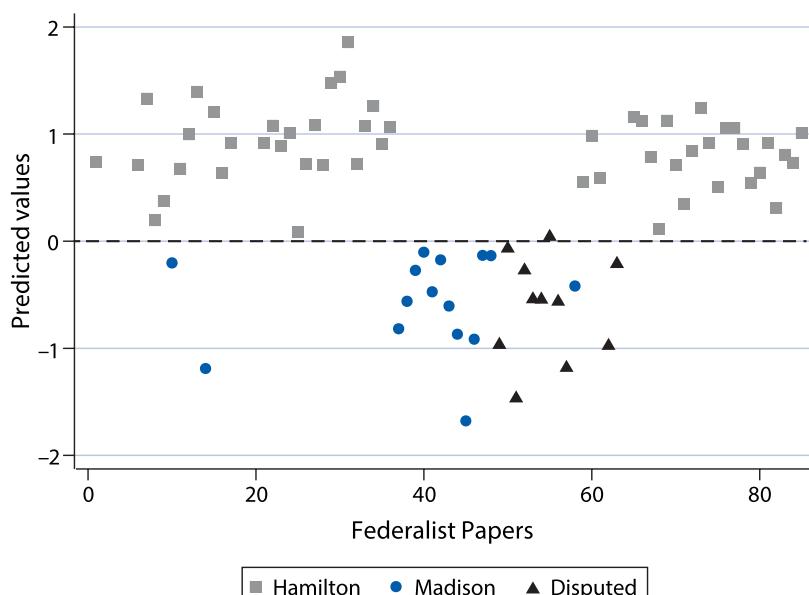
```
. list fedno authpred if inrange(fedno, 49, 57) | inrange(fedno, 62, 63)
```

	fedno	authpred
49.	49	-.967416
50.	50	-.0689504
51.	51	-1.468222
52.	52	-.2723972
53.	53	-.5423771

54.	54	-.5444193
55.	55	.0410413
56.	56	-.5643458
57.	57	-1.182824
62.	62	-.9769555
<hr/>		
63.	63	-.2121458

For ease of presentation, we plot the predicted values using different colors. Gray squares signify essays known to be written by Hamilton, while blue circles indicate those by Madison. Black triangles represent papers with disputed authorship. Points above (below) the dashed horizontal line, indicating zero, correspond to essays classified as written by Hamilton (Madison).

```
. scatter authpred fedno if hamilton == 1, msymbol(square) mcolor(red) || ///
> scatter authpred fedno if madison == 1, msymbol(O) mcolor(blue) || ///
> scatter authpred fedno if inrange(fedno, 49, 57) | ///
> inrange(fedno, 62, 63), msymbol(T) mcolor(black) ///
> yline(0, lpattern(dash) lcolor(black)) ///
> xtitle("Federalist papers") ytitle("Predicted values") ///
> legend(label(1 "Hamilton") label(2 "Madison") ///
> label(3 "Disputed") cols(3))
```



As our plots shows, the model predicts that Madison wrote all of the 11 essays except one. That one was barely classified as written by Hamilton, having a predicted value near zero.

## 7.4 Summary

This chapter introduced types of data different from those we analyzed in the previous chapters. We focused on how to discover systematic patterns in a variety of data. We began by analyzing network data. We visualized both **directed** and **undirected network data** with **graphs**, where nodes (or vertices) represent units, and edges between nodes represent relationships among them. We also showed how to compute various **centrality measures** in order to identify influential nodes within a given network. These measures include degree, closeness, and betweenness. These methods were illustrated through a classic application of the Florentine marriage network and a modern example of the Twitter following network among politicians.

Second, we considered **spatial** and **spatial-temporal data**. The spatial dimension split into two types: spatial point and spatial polygon data. We showed how maps can visualize spatial patterns effectively using John Snow's famous study of a cholera outbreak in 19th century London. Snow utilized a natural experiment to uncover the primary cause of the outbreak. We also used maps to visualize the outcome of the US presidential election and the diffusion of Walmart stores over time. Like the analysis of texts and networks, visualization plays a central role in spatial data analysis. To investigate how spatial patterns change over time, we created an **animation** that sequentially displayed a series of maps. This visualization effectively demonstrated the expansion of Walmart stores in the United States over the last several decades.

Finally, we analyzed **textual data** under the **bag-of-words assumption** that ignores the sequence of words. By focusing on the frequency of different terms within and across documents, we can discover topics that underlie the corpus. We introduced **term frequency-inverse document frequency** as a statistic that measures the importance of each term in a particular document. Using *The Federalist Papers* as an example, we also showed how the frequency of words can predict the authorship of essays with a linear regression model. To assess prediction accuracy while avoiding overfitting, we used cross validation (and in particular a leave-one-out cross validation procedure).

## 7.5 Exercises

### 7.5.1 INTERNATIONAL TRADE NETWORK

The size and structure of international trade flows varies significantly over time.<sup>13</sup> The volume of goods traded between countries has grown rapidly over the past century, as technological advances lowered the cost of shipping and countries adopted more liberal trade policies. At times, however, trade flows have decreased as a result of disruptive events such as major wars and the adoption of protectionist trade policies. In this exercise, we will explore

<sup>13</sup>This exercise is based in part on Luca De Benedictis and Lucia Tajoli. (2011), "The World Trade Network." *The World Economy*, vol. 34, no. 8, pp. 1417–1454. The trade data are from Katherine Barbieri and Omar Keshk. (2012). *Correlates of War Project Trade Data Set*, Version 3.0. Available at <http://correlatesofwar.org>

**Table 7.5.** International Trade Data.

Name	Description
country1	country name of exporter
country2	country name of importer
year	year
exports	total value of exports (in tens of millions of dollars)

The data are given for 1900, 1920, 1940, 1955, 1980, 2000, and 2009.

some of these changes by examining the network of international trade over several time periods. The data file `trade.dta` contains the value of exports from one country to another in a given year. The names and descriptions of variables in this data set are given in table 7.5. For the purposes of this exercise, we only include observations with two trading partners (i.e., `country1` and `country2` are nonmissing).

1. We begin by analyzing international trade as a directed network. For every year in the data set, create an adjacency matrix whose entry  $(i, j)$  equals country  $i$  exports to country  $j$ . If this export is zero, then the entry equals 0. We assume that missing data represent zero trade. Recode the `exports` variable accordingly. Plot the *network density*, which is defined over time as follows:

$$\text{network density} = \frac{\text{number of edges}}{\text{number of potential edges}}.$$

The `nwsummarize` command computes this measure, storing it in `r(density)`. Interpret the result.

2. Compute the measures of centrality based on degree, betweenness, and closeness (nonnormalized) for the years 1900, 1955, and 2009. The `nwload` command can load data from already-created networks, named in previous `nwset` commands. Consequently, it may be efficient to include this command when looping through years. For each year, list the five countries that have the largest values of these centrality measures. How do the countries on the lists change over time? Briefly comment on the results.
3. We now analyze the international trade network as a weighted, directed network in which each edge has a nonnegative weight proportional to its corresponding trade volume. For the years 1900, 1955, and 2009, instead of computing degree, compute the *graph strength*, which in this case equals the sum of imports and exports with all adjacent nodes. The `nwdegree` command with the `valued` option can be used to compute this weighted version of degree.<sup>14</sup>

<sup>14</sup>The `nwcommands` package does not compute other closeness measures with weighted networks.

**Table 7.6.** County-Level US Presidential Election Data.

Variable	Description
state	Full name of the 48 states (excluding Alaska, Hawaii, and the District of Columbia)
county	County name
year	Election year
rep	Popular votes for the Republican candidate
dem	Popular votes for the Democratic candidate
other	Popular votes for other candidates

Do the results differ from those of the unweighted network? Examine the top five countries. Can you think of another way to calculate centrality in this network that accounts for the value of exports from each country? Briefly discuss.

4. Apply the PageRank algorithm to the weighted trade network separately for 1900, 1955, and 2009. Identify the five most influential countries according to this algorithm within each year. In addition, examine how the ranking of PageRank values has changed over time for each of the following five countries: United States, United Kingdom, Russia, Japan, and China. Briefly comment on the patterns you observe.

### 7.5.2 MAPPING US PRESIDENTIAL ELECTION RESULTS OVER TIME

The partisanship of many states have been stable over time. For example, Massachusetts is a solidly “blue” state, having pledged its electoral votes to the Democratic candidate in 8 out of the last 10 presidential elections. On the other extreme, Arizona’s electoral votes went to the Republican candidate in 9 of the same 10 elections. Still, geography can occasionally be a poor predictor of presidential elections. For instance, in 2008, typically red states—including North Carolina, Indiana, and Virginia—helped elect Barack Obama to the presidency.

In this exercise, we will again map the US presidential election results for 48 states. However, our data will be more detailed in two respects. First, we will analyze data from 14 presidential elections from 1960 to 2012, allowing us to visualize how the geography of party choice has changed over time. Second, we will examine election results at the county level, allowing us to explore the spatial distribution of Democratic and Republican voters within states. Load the data file `elections.dta`. Each row of the data set represents the distribution of votes in that year’s presidential election from each county in the United States. Table 7.6 presents the names and descriptions of variables in this data set. The shapefile is too big for the QSS website but it can be downloaded at <https://catalog.data.gov/> (<https://dataset/tiger-line-shapefile-2016-nation-u-s-current-county-and-equivalent-national-shapefile>).

1. First, prepare the data. Use the `shp2dta` command to convert the `t1_2016_us_county` files into Stata format. Merge the county database with the US state database

created earlier, `usdb.dta`. Then, merge this new data set into `elections.dta` data. Keep only matched observations. To successfully merge the files, it will be necessary to clean the data so values of `NAME` are consistent across data sets. This may include, for example, using the `proper()` function to convert all names to the same case, the `subinstr()` function to ensure that "St." and "St" are consistently formatted, and the `regexm()` function to make the distinction between counties and cities that share the same name.

2. We next visualize the outcome of the 2008 US presidential election at the county level. Begin with Massachusetts and Arizona and visualize the county-level outcome by creating a basic choropleth map, coloring counties based on the Democratic two-party vote share as done in section 7.2.3. Ensure that the data are appropriately sorted before making your map. Use the Blues color scheme in the `spmap` command. Provide a brief comment.
3. Visualize the 2008 county-level election results across the United States as a whole.
4. We now examine how the geographical distribution of US presidential election results has changed over time at the county level. Starting with the 1960 presidential election, which saw Democratic candidate John F. Kennedy prevail over Republican candidate Richard Nixon, use animation to visualize the county-level election returns for each presidential election up to 2012. Base your code on what you programmed to answer the previous question. Remember that if you use `ffmpeg` to animate the graphs, they have to be numbered consecutively. Note that this task can take several minutes or longer, depending on processing capacity.
5. In this exercise, we quantify the degree of partisan segregation for each state. We consider a state to be politically segregated if Democrats and Republicans tend to live in different counties. A common way to quantify the degree of residential segregation is to use the *dissimilarity index* given by

$$\text{dissimilarity index} = \frac{1}{2} \sum_{i=1}^N \left| \frac{d_i}{D} - \frac{r_i}{R} \right|.$$

In the formula,  $d_i$  ( $r_i$ ) is the number of Democratic (Republican) votes in the  $i$ th county and  $D$  ( $R$ ) is the total number of Democratic (Republican) votes in the state.  $N$  represents the number of counties. This index measures the extent to which Democratic and Republican votes are evenly distributed within states. It can be interpreted as the percentage of one group that would need to move in order for its distribution to match that of the other group. Using data on Democratic and Republican votes from the 2008 presidential election, calculate the dissimilarity index for each state. Which states are among the most (least) segregated according to this measure? Visualize the result as a map. Briefly comment on what you observe. Use the `Greens2` color scheme and the `uscoord` data in the `spmap` command.

6. Another way to compare political segregation across states is to assess whether counties within a state are highly unequal in terms of how many Democrats or Republicans they have. For example, a state would be considered segregated if it had many counties filled with Democrats and many with no Democrats at all. In chapter 3, we considered the Gini coefficient as a measure of inequality (see section 3.6.2). Calculate the Gini coefficient for each state based on the percentage of Democratic votes in each county. Give each county the same weight, disregarding its population size.

To compute the Gini index, download and use the user-written `ineqdeco` command. Add `by(stateid)` to the end of the command to calculate the coefficients by state. The Gini coefficients will then be stored as return objects in `r(gini_stateid)`. Using a loop, store these values for each state in a new variable called `gini`. Which states are in the top (bottom) five with the highest (lowest) values of the index? Visualize the result using a map. What is the correlation between the Gini index and the dissimilarity index you calculated in the previous question? How are the two measures conceptually and empirically different? Briefly comment on what you observe and explain the differences between the two indexes.

7. Lastly, we examine how the degree of political segregation has changed in each state over time. Use animation to visualize these changes. Briefly comment on the patterns you observe.

### 7.5.3 ANALYZING THE PREAMBLES OF CONSTITUTIONS

Some scholars argue that over the last centuries, the US Constitution has emerged, either verbatim or paraphrased, in numerous founding documents across the globe.<sup>15</sup> Will this trend continue, and how might one even measure constitutional influence, anyway? One way is to see which constitutional rights (such as free speech) are shared across the founding documents of different countries, and observe how this commonality changes over time. An alternative approach, which we take in this exercise, is to examine textual similarity among constitutions. We focus on the preamble of each constitution, which typically states the guiding purpose and principles of the rest of the constitution. Table 7.7 presents the names and descriptions of the constitution preambles in `constitution.dta`.

1. Let's begin by visualizing the data to better understand how constitutional documents differ. Start by importing the preamble data into Stata, and then preprocess the text. Create variables for both the regular document term frequency (`tf`), and for the weighted term frequency (`tf-idf`). In both cases, visualize the preamble to the US Constitution with a word cloud. Because our corpus is much smaller than that analyzed earlier in the chapter, it is necessary to multiply both frequencies by a larger number so their marker size is sufficiently visible (e.g., `replace freq = freq * 5`, `replace tfidfw = tfidfw * 70`). How do the results differ between the two methods? Note that we normalize the

<sup>15</sup>This exercise is in part based on David S. Law and Mila Versteeg (2012). “The declining influence of the United States constitution,” *New York University Law Review* vol. 87, no. 3, pp. 762–858, and Zachary Elkins, Tom Ginsburg, and James Melton. (2012). “Comments on law and Versteeg’s The Declining Influence of the United States Constitution.” *New York University Law Review* vol. 87, no. 6, pp. 2088–2101.

**Table 7.7.** The Names and Descriptions of Variables in the Constitution Preamble Data.

Name	Description
country	the country name with words separated by underscores
year	the year the constitution was created
preamble	raw text of the preamble to the constitution

The data set contains the raw textual information about the preambles of constitutions around the world.

tf-idf weights by dividing the term frequency by the total number of documents so that lengthy constitutions do not receive greater weights.<sup>16</sup>

2. We next apply the  $k$ -means algorithm to the rows of the tf-idf matrix and identify clusters of similar constitution preambles. Before reshaping the tf-idf variable into a matrix, we need to make each one comparable by dividing it by a constant such that each country represents a vector of unit length. Note that the length of a vector  $a = (a_1, a_2, \dots, a_n)$  is given by  $\|a\| = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$ .

To convert to unit length, square each tf-idf score (store as `tfidfw_sq`) and then sum the elements of each country (`egen tf_sum = sum(tfidfw_sq)`). Take the square root of this sum to obtain  $\|a\|$  for each tf-idf score (`generate tfunit = tfidfw / sqrt(tf_sum)`). The data can be reshaped to wide format so that each word's converted tf-idf score is a variable. But first, merge in the constitution data based on id number so we can identify the countries within each cluster. Save the data prior to using the `reshape` command for use in the remaining exercises. Then, conduct the  $k$ -means analysis with 5 random (`krandom`) starting centers, setting the seed to 12345, and describe the results. Due to version limitations on the maximum number of variables, some users may be unable to complete the analysis on the full set of data. In this case, users should focus on a subset of the data.

3. Now we will see whether foreign constitutions became more or less similar to the preamble of the US Constitution over time. In the document-term matrix, each document is represented as a vector of term frequencies. To compare two documents, we define *cosine similarity* as the cosine of the angle  $\theta$  between the two corresponding  $n$ -dimensional vectors,  $a = (a_1, a_2, \dots, a_n)$  and  $b = (b_1, b_2, \dots, b_n)$ . Formally, the measure is defined as

<sup>16</sup>The `wordcloud` command is as inclusive as possible when identifying words in the input file. Consequently, it also preserves words that pertain to the Stata file itself, including “stata,” “varnames,” “release,” “dta,” and so on. The inclusion of these words is inconsequential when dealing with large bodies of text, like that found earlier in this chapter. However, here, where the preambles are relatively short, these terms could show up as prominent. Therefore, the following words should be dropped prior to creating a wordcloud: `byteorder`, `data`, `dta`, `formats`, `label`, `labels`, `gso`, `lsf`, `names`, `preamble`, `release`, `sortlist`, `stata`, `strls`, `types`, `timestamp`, `variable`, `varnames`, and `value`. For the convenience of the reader, we provide a simple do-file called `stata_drop_wc` to drop these terms. See also Mehmet F. Dicle and Betul Dicle. (2018). “Content analysis: Frequency distribution of words.” *The Stata Journal* vol. 18, no. 2, pp. 379–386.

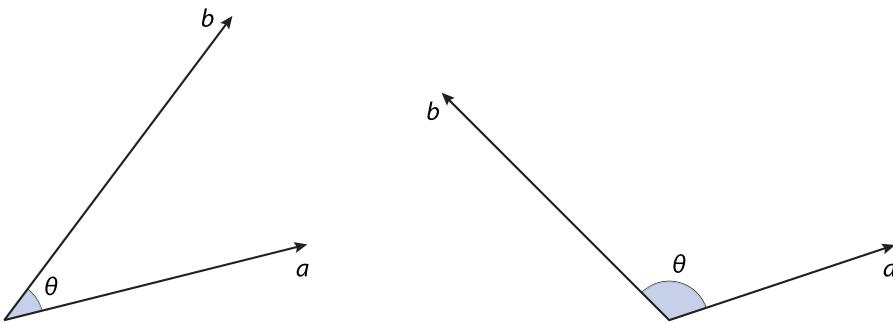


Figure 7.6. Cosine Similarity of Two Vectors. Two two-dimensional vectors  $a$  and  $b$  have a positive (negative) value of cosine similarity in the left (right) plot.

follows,

$$\text{cosine similarity} = \cos \theta = \frac{a \cdot b}{\|a\| \cdot \|b\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}.$$

The numerator represents the so-called *dot product* of  $a$  and  $b$ , while the denominator is the product of the lengths of the two vectors. The measure ranges from  $-1$  (when the two vectors go in the opposite directions) to  $1$  (when they completely overlap). As illustrated in figure 7.6, two vectors have a positive (negative) value of cosine similarity when they point in similar (different) directions. The measure is zero when they are perpendicular to each other.

The **lsemantica** package, available by typing `net search lsemantica`, offers advanced features for examining similarity in text documents, including cosine similarity as defined above using its `lsemantica_cosine` command. Begin by reshaping the data we saved in question 2 from long into wide format, so that the weighted tf-idf scores for each word are stored in columns. Remember to replace missing weighted tf-idf values with 0.

4. Once you have reshaped the data, use the `lsemantica_cosine` command to calculate the cosine similarity for all documents. Because we have not reviewed this package in the chapter, we list the full command.

```
. lsemantica_cosine tfidfw*, mean_cosine min_cosine max_cosine find_similar(5)
  find_similar_cosine(5)
```

This command will provide the five most similar documents for each document. The `find_similar` option provides a set of variables `most_similar_*`, which sequentially store the index number (`_n`) of the most similar documents. For example, `most_similar_4` will have the index number of the country who has the fourth

most similar document. The `find_similar_cosine` option then returns the cosine similarity values that correspond to these rankings, storing them in variables `cosine_most_similar_*`.

To illustrate, we look at the five countries with preambles most similar to the first country (`_n == 1`) in our data set, Afghanistan.

```
. list country if _n==1
```

country	
1.	afghanistan

```
. list most_similar_* cosine_most_similar_* if _n == 1
```

1.	most_s~1 65	most_s~2 67	most_s~3 147	most_s~4 46	most_s~5 81	cosine~1 .09940685	cosine~2 .09458501	cosine~3 .09047615
						cosine~4 .08848946	cosine~5 .08484786	

```
. list country if inlist(_n, 65, 67, 147, 46, 81)
```

country	
46.	ethiopia
65.	iran_islamic_republic_of
67.	ireland
81.	libya
147.	united_arab_emirates

Because `lsemantica_cosine` returns the indexing numbers of the countries, we have to create a second list to display the country names, using the values in the `most_similar_*` variables. We can see that the Islamic Republic of Iran shares the highest similarity with the preamble of the Afghan constitution. Now find the five constitutions whose preambles most resemble that of the US Constitution. The US is indexed at 149 (`_n == 149`).

- Finally, examine the influence of the US Constitution on other constitutions over time. We focus on the postwar period. Calculate the cosine similarity score between the United

States and all other countries. Replacing 5 with 154 in the command line of the previous question will provide a separate similarity score between each country, not just the top five most similar. It will be necessary to reshape the data from wide to long format, and to remerge the constitution data to fill in the constitution years of the comparison countries (whose id numbers will be saved in reshaped variable `most_similar`\_ that correspond to the similarity scores saved in `cosine_most_similar`.<sup>17</sup>

For every 10 years from 1950 until 2010, calculate the average of cosine similarity between the US Constitution and the constitutions that were created during each decade. Plot the result. Each of these averages computed over time is called the *moving average*. Does similarity tend to increase, decrease, or remain the same over time? Comment on the pattern you observe.

<sup>17</sup>These calculations can also be made using the `matrix dissimilarity` command with the `angular` option in Stata, where users must create a loop that compares each country's row of weighted tf-idf scores to the US row, storing each similarity score in a variable to then plot (see `help matrix dissimilarity` for more information).

## Chapter 8

---

# Next

Statistics are no substitute for judgment.  
—Henry Clay

What comes next? There are several directions one could take to further improve data analysis skills. The current book is a first course in applied data analysis and introduces only a tiny fraction of useful data analytic methods. There is much more to learn. An obvious next step is to learn more about data analysis and statistics. For example, one might enroll in a second course in data analysis and statistics (or read a relevant textbook) that covers regression modeling techniques, which are essential tools for quantitative social science. Another possibility is to take a course on specific topics of interest, such as causal inference, social network analysis, and survey methodology.

As an introduction to quantitative social science, this book does not take a mathematical approach to data analysis. Instead, the focus of the book is to give readers a sense of how data analysis is used in quantitative social science research, while teaching elementary concepts and methods. But since all of data analysis and statistical methods have a mathematical foundation, a deeper understanding of them requires a good command of mathematics. A better grasp of methods will, in turn, enable one to become a more sophisticated user of data analysis and statistics who can critically assess the advantages and limitations of various methodologies in applied research. Furthermore, if one is interested in becoming a methodologist who develops new methods, a solid foundation in mathematics is critical. In particular, it is essential to learn multivariate calculus and linear algebra, followed by probability theory. After these foundations, students can learn statistical theory and various modeling strategies in a rigorous fashion.

Since the main focus of this book is data analysis, we did not discuss how to collect data—yet without data collection, there would be no data analysis. Although we analyzed the data from several randomized controlled trials in this book, little attention was given to experimental designs. How should we recruit subjects when conducting an experiment? What are the experimental design strategies one could use to obtain precise estimates of causal effects? These and other questions arise when designing experiments in the laboratory and field. A pioneer statistician, Ronald A. Fisher, once stated, “To call in the statistician after the experiment is done may be no more than asking him to perform a postmortem

examination: he may be able to say what the experiment died of.”<sup>1</sup> We must learn how to design randomized experiments in order to take advantage of this powerful tool for causal inference. Even for observational studies, careful planning is required to identify the instances in which researchers can draw causal inference in a credible manner. Research design forms a fundamental component of quantitative social science research.

Similarly, while we analyze survey data in this book, we do not examine survey sampling strategy and questionnaire design. In many cases, the simple random sampling we discussed is not feasible, because we do not have a sampling frame that contains a complete list of all individuals of a target population. For example, when studying a population that is difficult to reach (e.g., homeless people, seasonal migrants), other strategies, such as respondent-driven sampling, have been used. Another important question is how to correct for the lack of representativeness in survey data. In particular, Internet surveys are now commonly used, but an online panel is often far from being representative of a target population. Questionnaire design also plays an essential role in obtaining accurate measurements. In chapter 3, we saw examples of a special technique for eliciting truthful answers to sensitive questions. The exercise in section 3.9.2 introduced a survey methodology that reduces measurement error due to the possibility that respondents may interpret the same questions differently. These examples suggest that studying a variety of data collection strategies is as important for quantitative social scientists as learning about data analysis.

While different interests may take people in various directions after completing this book, everyone should continue to practice data analysis. In the words of John W. Tukey,<sup>2</sup> “If data analysis is to be helpful and useful, it must be practiced.” Now that users of this book have learned the basic methodology and programming necessary for data analysis, they should begin to conduct quantitative social science research by analyzing data sets of their choice. Just as with data analysis, one learns how to conduct research only by doing, not by reading the research of other people. With the massive amount of data available online, anyone from undergraduate to graduate students and from practitioners to academic researchers should be able to start making their own data-driven discoveries.

This book highlights the power of data analysis. However, it is also important to be aware of its fundamental limitations when analyzing data. In particular, data analysis is far from objective. Good data analysis must be accompanied by sound judgment, which is in turn built upon one’s knowledge and experience. Without substantive theories, data analysis can be easily misguided. In quantitative social science research, we analyze data for the purpose of better understanding society and human behavior. This goal is unattainable unless we use social science theories to determine how data should be analyzed. Stronger theoretical guidance is required for the analysis of “big data” because without it we will not know where to look for interesting patterns.

Although a solid grasp of the mathematics that underlie statistical theories and methods is important, we should not underestimate the value of contextual knowledge about the data sets to be analyzed. For example, to competently design and analyze the survey of Afghan civilians introduced in chapter 3, researchers had to understand the cultural, political, and economic environment of local communities in Afghanistan where the respondents live.

<sup>1</sup>Ronald A. Fisher (1938). “Presidential address: The first session of the Indian Statistical Conference, Calcutta, 1938.” *Sankhyā*, vol. 4, pp. 14–17.

<sup>2</sup>John W. Tukey (1962). “The future of data analysis.” *Annals of Mathematical Statistics*, vol. 33, no. 1, pp. 1–67.

Interviewing individuals who have little education, during a civil war, is a challenging task. The researchers worked with a local survey firm to gain access to rural villages through negotiation with local leaders and militants. For cultural reasons, they were unable to interview female respondents, and interviews had to take place in a public sphere where village elders were able to listen to survey questions and answers. Randomized response methodology is a classic survey method for asking sensitive questions while protecting the secrecy of individual responses. However, this method was seen as inappropriate in the study because the required randomization using coins or dice was considered to be against Islamic law. Other challenges in this study included how to ask respondents' tribal affiliation, how to measure the level of wealth when the economy is largely informal, and what policy questions to ask when measuring respondents' political ideology.

These examples illustrate the importance of contextual knowledge in designing and implementing quantitative social science research. Therefore, data analysts should learn about the relevant substance and background of their study, either on their own or by partnering with experts, well before starting to analyze data. They should also be aware of the danger that mechanical applications of statistical methods to data may lead to unreliable empirical findings. Indeed, this is the reason why applied statistics has developed separately in a variety of fields of the natural and social sciences. While statistical methods rest on universal mathematical theory and are widely applicable, their application requires specific substantive knowledge. The goal of this book has been to illustrate this unique feature of data analysis and statistics by showing how general methods can be used to answer interesting social science questions.

With rapid advancements in technology and data availability, the world needs those who can creatively combine substantive knowledge with data analysis skills in all fields, from academia to journalism. This book opens the door to this exciting world of data analysis.



# General Index

---

## SYMBOLS

$R^2$ , 350  
 $z$ -score, 113, 157, 251, 253, 269

## A

absolute value, 73  
addition rule, 200  
adjacency matrix, 365  
adjusted  $R^2$ , 176, 350  
age-specific death rate, 31  
age-specific fertility rate, 29  
alternative hypothesis, 315  
AND, 39  
animation, 398  
association, 50  
asymptotic theorems, 264  
average treatment effect, 50  
axioms, 200

## B

bag-of-words, 406  
bar plot, 87  
Bayes' rule, 227  
Bayesian, 198  
before-and-after design, 63  
Bernoulli random variable, 241  
betweenness, 372, 378  
bias, 135, 278  
bin, 89  
binary random variable, 241  
binary variable, 14, 37  
binomial distribution, 246  
binomial theorem, 248  
birthday problem, 203  
box plot, 92  
butterfly ballot, 164

## C

categorical variable, 42  
causal effects, 47  
causal inference, 47  
ceiling effects, 104  
census, 96

centering, 117  
central limit theorem, 266, 290, 306, 330, 348, 351  
centrality, 368  
centroid, 117  
ceteris paribus, 171  
classification, 139  
classification error, 417  
closeness, 370, 378  
clustering algorithms, 117  
clusters, 117  
coefficient of determination, 160, 176, 350  
coefficients, 148  
combinations, 208  
complement, 201  
complete randomization, 279, 308  
computational revolution, 1  
conditional cash transfer program, 193  
conditional expectation, 337  
conditional expectation function, 338  
conditional independence, 222, 235  
conditional probability, 213, 215  
confidence bands, 291  
confidence interval, 291  
confidence level, 291  
confounders, 60, 387  
confounding bias, 60  
confusion matrix, 139  
consistent, 278  
contingency table, 19  
continuous random variable, 241, 242  
control group, 50, 57  
correlation, 112, 147  
correlation coefficient, 112  
cosine similarity, 425  
counterfactual, 47  
covariance, 351  
coverage probability, 293  
critical value, 291, 306  
cross-section comparison design, 56  
cross-section data, 63  
cross-tabulation, 19  
crude birth rate, 28  
crude death rate, 30

- cumulative distribution function (CDF), 242, 243  
 cumulative sum, 265
- D**  
 data revolution, 1  
 data-generating process, 148, 206, 281, 336  
 decile, 71  
 degree, 368, 376  
 degrees of freedom, 176, 288, 305  
 density, 89, 243  
 descriptive statistics, 67  
 dichotomous, 14  
 difference-in-differences, 64  
 difference-in-means estimator, 50, 169, 279  
 directed network, 366, 375  
 discrete random variable, 241  
 dissimilarity index, 423  
 disturbance, 148  
 document frequency, 409  
 document-term matrix, 411  
 dot product, 426  
 dummy variable, 14  
 DW-NOMINATE scores, 106
- E**  
 ecological inference, 361  
 edges, 367  
 Electoral College, 128  
 error, 148  
 error bands, 291  
 estimation error, 277  
 estimator, 276  
 event, 199  
 exogeneity, 337  
 expectation, 256, 278  
 experiment, 199  
 experimental data, 33  
 exploratory data analysis, 364  
 external validity, 51, 56, 189
- F**  
 factor, 42  
 factor variable, 42, 87  
 factorial, 202  
 factorial variable, 42, 87  
 false discoveries, 327  
 false discovery rate, 232  
 false negative, 139  
 false positive, 139, 232  
 farness, 370  
 file drawer bias, 359  
 first moment, 258  
 first quartile, 70  
 Fisher's exact test, 315  
 fitted value, 148  
 floor effects, 104  
 frequentist, 197  
 function, 10  
 fundamental problem of causal inference, 48, 308
- G**  
 Gaussian distribution, 249  
 get-out-the-vote, 51  
 Gini coefficient, 109  
 Gini index, 109  
 Google, 381  
 graph, 367  
 graph strength, 421
- H**  
 Hawthorne effect, 52, 55  
 heterogeneous treatment effects, 177  
 heteroskedastic, 348  
 heteroskedasticity-robust standard errors, 348  
 histogram, 89, 136  
 homoskedasticity, 346  
 hypothesis testing, 307
- I**  
 i.i.d., 246  
 ideology, 105  
 idf, 409  
 if qualifier, 38  
 immutable characteristics, 49  
 in-sample prediction, 167, 417  
 indegree, 377  
 independence, 218  
 independently and identically distributed, 246  
 indicator, 171  
 indicator function, 263  
 Institutional Review Board, 103  
 integration, 257  
 interaction effect, 178  
 intercept, 148  
 internal validity, 51, 56, 188, 189  
 interquartile range (IQR), 70  
 inverse document frequency, 409  
 inverse function, 187  
 item count technique, 103  
 item nonresponse, 101  
 item response theory, 106  
 iterations, 131  
 iterative algorithm, 117
- J**  
 joint independence, 222  
 joint probability, 214
- K**  
 Kish grid, 99
- L**  
 large sample theorems, 264  
 law of iterated expectation, 345

law of large numbers, 264, 278, 281  
 law of total probability, 201, 211, 214, 222  
 law of total variance, 346  
 least squares, 151  
 leave-one-out cross validation, 417  
 level of test, 314  
 limit, 197  
 linear model, 148  
 linear regression, 144  
 linear relationship, 147  
 list experiment, 103  
 logarithmic transformation, 99, 204  
 logical conjunction, 39  
 logical disjunction, 39  
 logical operators, 39  
 longitudinal data, 63  
 longitudinal study, 75  
 loop, 131, 418  
 Lorenz curve, 109  
 lower quartile, 70

**M**

macros, 129  
 maps, 386  
 margin of error, 296  
 marginal probability, 212  
 matrices, 219  
 mean-squared-error, 286  
 measurement models, 105  
 median, 67, 91  
 misclassification, 139  
 misreporting, 102  
 Monte Carlo error, 207, 284  
 Monte Carlo simulation, 205, 225, 245, 260, 265, 281  
 Monty Hall problem, 224  
 moving average, 191  
 multiple testing, 327  
 multistage cluster sampling, 98

**N**

natural experiment, 79, 386, 387  
 natural logarithm, 99  
 nearness, 370  
 network data, 364  
 network density, 421  
 no omitted variables, 339  
 nodes, 367  
 nonlinear relationship, 148  
 nonresponse, 278  
 normal distribution, 249, 266  
 null hypothesis, 313  
 numeric variable, 89

**O**

observational studies, 56, 338  
 one-sample *t*-test, 319  
 one-sample *z*-test, 319

one-sample tests, 316  
 one-sided *p*-values, 315  
 one-tailed *p*-values, 315  
 OR, 39  
 out-of-sample prediction, 167, 417  
 outcome variable, 33  
 outdegree, 377  
 outliers, 67, 122, 164  
 overfitting, 167, 417

**P**

packages, 23  
 PageRank, 381  
 panel data, 63, 141  
 parameter, 276  
 Pascal's triangle, 248  
 percentile, 71  
 permutations, 202  
 person-year, 28  
 placebo test, 188  
 political polarization, 109  
 polity score, 79  
 population average treatment effect, 280  
 population mean, 256  
 positive predictive value, 227  
 posterior probability, 227  
 potential outcomes, 48  
 power, 329  
 power analysis, 329  
 power function, 332  
 predicted value, 148  
 prediction error, 135, 148  
 pretreatment variables, 55, 60  
 prior probability, 227  
 probability, 197  
 probability density function (PDF), 243  
 probability distributions, 241  
 probability mass function (PMF), 242  
 probability model, 241  
 probability sampling, 96  
 Progresa, 193  
 proof by contradiction, 313  
 publication bias, 327, 359

**Q**

Q–Q plot, 115, 123, 253, 306  
 quadratic function, 182  
 quantile–quantile plot, 115, 123, 253, 306  
 quantile treatment effects, 76  
 quantiles, 67, 71, 115  
 quartiles, 70  
 quincunx, 267  
 quintile, 71  
 quota sampling, 97

**R**

random digit dialing, 98  
 random variables, 241

- randomization inference, 313  
 randomized controlled trials, 49, 279, 338  
 randomized experiments, 49  
 randomized response technique, 104  
 rational number, 313  
 receiver, 366  
 reference distribution, 313  
 regression discontinuity design, 185  
 regression line, 148  
 regression toward the mean, 154, 253  
 relational operators, 38  
 representative, 97  
 residual, 148, 171  
 residual plot, 163  
 residuals, 253  
 root mean square (RMS), 72, 136, 152  
 root-mean-squared error, 136, 152, 286,  
     350  
 rule of thumb, 296
- S**
- sample average treatment effect, 50, 279  
 sample average treatment effect for the treated,  
     65  
 sample correlation, 351  
 sample mean, 59, 256  
 sample selection bias, 51, 97  
 sample size calculation, 297  
 sample space, 199  
 sampling distribution, 278, 287, 313  
 sampling frame, 97, 98, 101  
 sampling variability, 260  
 sampling with replacement, 205  
 sampling without replacement, 97, 206  
 scalar, 59, 129  
 scaling, 117  
 scatterplot, 106, 145  
 scientific significance, 316, 320  
 scraping, 401  
 second moment, 258  
 second quartile, 70  
 selection bias, 61  
 selection on observables, 339  
 sender, 366  
 set, 199  
 sharp null hypothesis, 313  
 simple random sampling, 96, 206, 277  
 simple randomization, 279, 308  
 simulation, 205  
 slope, 148  
 social desirability bias, 27, 102  
 sparse, 411  
 spatial data, 386  
 spatial point data, 386  
 spatial polygon data, 386, 389  
 spatial voting, 105  
 spatial-temporal data, 386  
 standard deviation, 72, 73, 258  
 standard error, 287
- standard normal distribution, 250, 253, 281  
 standardize, 117  
 standardized residuals, 253  
 statistical control, 61  
 statistical significance, 316, 320  
 step function, 247  
 Student's *t*-distribution, 304  
 Student's *t*-test, 333, 335  
 subclassification, 61  
 sum of squared residuals, 151, 171  
 supervised learning, 121, 407  
 support, 257  
 survey, 82  
 survey sampling, 96
- T**
- tercile, 71  
 term frequency, 404, 406, 409  
 term frequency-inverse document frequency,  
     409
- term-document matrix, 411  
 test statistic, 313  
 tf, 404  
 tf-idf, 409  
 third quartile, 70  
 time trend, 64  
 time-series, 141  
 time-series operators, 141  
 time-series plot, 108, 143  
 topics, 406  
 total fertility rate, 29  
 total sum of squares, 160  
 treatment, 48  
 treatment group, 50, 56  
 treatment variable, 33, 48  
 true positive rate, 227, 231  
 true positives, 231  
 two-sample *t*-test, 323  
 two-sample *z*-test, 322  
 two-sample tests, 316  
 two-sided *p*-value, 315, 317  
 two-tailed *p*-value, 315  
 type I error, 314  
 type II error, 314, 329
- U**
- unbiased, 136, 278  
 unconfoundedness, 339  
 uncorrelated, 337  
 undirected network, 366, 375  
 uniform random variable, 242  
 unit nonresponse, 101, 301  
 unobserved confounders, 338  
 unsupervised learning, 121, 407  
 upper quartile, 70
- V**
- value labels, 16

variable labels, 16  
variables, 10  
variance, 74, 258  
Venn diagram, 200, 201  
vertices, 367

**W**

weighted average, 236  
with replacement, 97  
word cloud, 406  
working directory, 20



# Stata Index

---

## SYMBOLS

\* /, 26  
/\*, 26  
//, 26  
///, 26  
<, 38  
<=, 38  
=, 38  
==, 38  
>, 38  
>=, 38  
&, 39  
\_b, 175  
\_b[\_cons], 150  
\_b[varname], 150  
\_cons, 150  
\_n, 15

## A

addplot(), 253  
append, 21, 23

## B

binomial(), 247, 262  
binomialp(), 246, 262  
binomialtail(), 262  
browse, 21, 34, 46  
bsample, 206  
by, 216  
bysort, 28, 108

## C

cd, 20  
centile, 71, 126  
ci, 302  
clear, 11  
cluster kmeans, 118  
codebook, 13, 18, 53, 57  
coeflegend, 150  
collapse, 45, 46, 110, 126  
color(), 89  
comb(), 210, 310

combomarginsplot, 353  
cond(), 40, 58  
convert, 400  
correlate, 114, 147  
count, 217

## D

date(), 134  
describe, 12, 13, 33, 57  
destring, 12  
display, 9  
do, 25  
drop, 38, 44, 45  
duplicates drop, 173

## E

e(), 152  
edit, 21, 34, 46  
egen, 43, 73, 108, 117, 157, 231, 253,  
    366  
encode, 340  
end, 11  
ereturn list, 176  
estadd, 354  
estimates replay, 349  
estimates restore, 349  
estimates store, 340, 349  
estout, 219  
estpost, 219, 223  
exp(), 100, 187, 204

## F

ffmpeg, 399  
ffmpeg package, 423  
fileread, 402  
findit, 24  
for, 133  
force, 85  
foreach, 131, 132  
forvalues, 131, 132  
fs, 401  
fs package, 401, 402

**G**

generate, 14, 34, 58, 73, 101, 108, 127, 265  
 generate(), 16  
 global, 130  
 graph, 94  
 graph bar, 87, 89  
 graph box, 92, 93  
 graph combine, 95  
 graph dir, 95  
 graph display, 95  
 graph export, 398  
 graph save, 94  
 graph use, 95  
 grmap package, 389  
 group(), 43, 216  
 gsort, 23, 377, 392, 409

**H**

help, 10, 43, 89  
 histogram, 89, 90, 123, 136

**I**

if, 41, 44  
 if command, 207  
 import delimited, 21  
 inlist(), 16, 127  
 input, 11  
 invnormal(), 274, 292, 306  
 invt(df, p), 306

**J**

joinby, 23

**K**

keep, 38, 44, 45

**L**

label define, 16, 17  
 label values, 17  
 levelsof, 231, 407  
 lfit, 151, 186  
 line, 108, 151  
 list, 12, 14, 34, 164  
 lnfactorial(), 204  
 local, 130  
 log(), 100  
 log10(), 100  
 lookfor, 18  
 loop, 131  
 lpattern(), 89  
 ls, 20  
 lsemantica package, 426  
 lsemantica\_cosine, 426

**M**

macro dir, 130  
 margins, 181–183, 352, 355

marginsplot, 184, 353  
 mcolor(), 89  
 merge, 21, 111, 230, 237  
 missing, 89, 212  
 missings, 85, 86  
 missings package, 85, 86  
 mlabel(), 137  
 mlabposition(), 137  
 modes, 35  
 modes package, 24  
 mszie(), 89  
 msymbol(), 89

**N**

net search, 24  
 nolabel, 42  
 normal(), 251, 255, 274, 317  
 nwbetween, 372, 379  
 nwcloseeness, 370  
 nwcommands, 421  
 nwcommands package, 367  
 nwcommands-ado package, 367  
 nwdegree, 368, 377, 421  
 nwplot, 367  
 nwset, 367, 375, 385  
 nwsummarize, 421

**O**

over(), 89

**P**

power, 333–335  
 power onemean, 335  
 predict, 150–152, 160, 173, 418  
 preserve, 45, 46, 162, 173  
 program, 269  
 prtest, 320–322, 324, 325, 327  
 prtesti, 320  
 pwcorr, 122

**Q**

qnorm, 253  
 qqplot, 115  
 quietly, 25

**R**

rbinomial(), 260  
 recode, 15  
 regress, 149, 157, 162, 171, 173, 339, 340, 342, 349, 352  
 relabel(), 89  
 replace, 21, 41, 127  
 reshape, 411  
 residuals, 150  
 restore, 45, 46, 162, 173  
 return list, 54, 80

rowtotal(), 142  
runiform(), 245, 265, 282  
runiformint, 225  
rvfplot, 163

**S**  
sample, 206  
scalar(), 59  
scatter, 106, 146, 299, 407  
scatteri, 138  
scheme(), 89  
set maxvar, 411  
shell, 399  
shp2dta package, 389  
sign(), 138  
simulate, 270, 282, 284, 311  
sort, 15, 23, 47  
spmap, 391, 393, 394, 396, 397  
spmap package, 389  
spshape2dta package, 389  
ssc install, 24  
std(), 157  
sum(), 265  
summarize, 18, 38, 43, 45, 54, 70, 74,  
    82, 159, 164  
summarize(), 19, 54  
svmat, 221  
swirl package, xx  
syntax, 269

**T**  
tab1, 35  
table, 55  
tabstat, 55, 58, 70, 74, 102, 118

tabulate, 18, 19, 35, 37, 41–43, 46, 54, 55,  
    61, 80, 82–84, 87, 212–214, 315  
tempfile, 131, 156  
text, 91  
title(), 88  
tset, 141  
ttest, 306, 322, 324  
twoway line, 89  
twoway rspike, 299  
twoway scatter, 89, 120  
txttool, 403, 413  
txttool package, 403

**U**  
use, 20, 33, 53, 82  
usespss, 21  
ustrregrexra(), 403

**W**  
while(), 382, 383  
winexec, 399  
wordcloud package, 403

**X**  
xb, 150  
xlabel(), 88  
xline(), 91  
xtitle(), 88

**Y**  
ylabel(), 88  
yline(), 91  
ytitle(), 88



# Stata Command Abbreviation List

---

## A

append ..... app

## B

browse ..... br  
bysort ..... bys

## C

centile() ..... c()  
cluster kmeans ..... cluster k  
coeflegend ..... coefl  
color() ..... c()  
correlate ..... corr,cor  
count ..... cou

## D

describe ..... d  
display ..... di

## E

encode ..... en  
ereturn list ..... eret  
estimates replay ..... estr  
estimates restore ..... etsres  
estimates store ..... eststo

## F

forvalues ..... forv

## G

generate ..... gen,g  
global ..... gl  
graph ..... gr

## H

help ..... h  
histogram ..... hist

## I

import delimited.....import delim  
input ..... inp

## L

label define ..... la de  
label values ..... la val  
label variable ..... la var

local ..... loc  
lcolor() ..... lc  
lpattern() ..... lp

## M

macro dir ..... ma di  
mcolor() ..... mc()  
merge ..... mer  
mlabel() ..... mlab(),ml()  
mlabposition() ..... mlabp()  
msize() ..... msiz()  
msymbol() ..... msym(),m()

## N

nolabel ..... nol

## P

program ..... pr

## Q

quietly ..... qui

## R

regress ..... reg  
return list ..... ret li

## S

scalar ..... sca  
scatter ..... sc  
set seed ..... set se  
shell ..... sh  
summarize ..... sum,su

## T

tabulate ..... tab,ta  
title() ..... ti()  
twoway line ..... tw line  
twoway scatter ..... tw sc

## X

xlabel() ..... xlab()  
xtitle() ..... xti()

## Y

ylabel() ..... ylab()  
ytitle() ..... yti()

