

# Microeometrics Using Stata

Volume I: Cross-Sectional and  
Panel Regression Methods

Second Edition



A. COLIN CAMERON  
PRAVIN K. TRIVEDI

STATA  
Press

# Microeometrics Using Stata

Volume I: Cross-Sectional and  
Panel Regression Methods

Second Edition



A. COLIN CAMERON  
PRAVIN K. TRIVEDI

STATA  
Press

# Microeometrics Using Stata

## Volume I: Cross-Sectional and Panel

### Regression Methods

Second Edition

A. COLIN CAMERON

*Department of Economics*

*University of California, Davis, CA*

*and*

*School of Economics*

*University of Sydney, Sydney, Australia*

PRAVIN K. TRIVEDI

*School of Economics*

*University of Queensland, Brisbane, Australia*

*and*

*Department of Economics*

*Indiana University, Bloomington, IN*



A Stata Press Publication

StataCorp LLC

College Station, Texas



Copyright © 2009, 2010, 2022 by StataCorp LLC  
All rights reserved. First edition 2009  
Revised edition 2010  
Second edition 2022

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas  
77845

Typeset in LaTeX2e

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Print ISBN-10: 1-59718-359-8 (volumes I and II)

Print ISBN-10: 1-59718-361-X (volume I)

Print ISBN-10: 1-59718-362-8 (volume II)

Print ISBN-13: 978-1-59718-359-8 (volumes I and II)

Print ISBN-13: 978-1-59718-361-1 (volume I)

Print ISBN-13: 978-1-59718-362-8 (volume II)

ePub ISBN-10: 1-59718-360-1 (volumes I and II)

ePub ISBN-10: 1-59718-363-6 (volumes I)

ePub ISBN-10: 1-59718-364-4 (volumes II)

ePub ISBN-13: 978-1-59718-360-4 (volumes I and II)

ePub ISBN-13: 978-1-59718-363-5 (volumes I)

ePub ISBN-13: 978-1-59718-364-2 (volumes II)

Library of Congress Control Number: 2022938057

No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means—electronic, mechanical, photocopy, recording, or otherwise—with the prior written permission of StataCorp LLC.

Stata, **stata**, Stata Press, Mata, **mata**, and NetCourse are registered trademarks of StataCorp LLC.

Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations.

NetCourseNow is a trademark of StataCorp LLC.

LaTeX2e is a trademark of the American Mathematical Society.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# **Contents**

## **Preface to the Second Edition**

## **Preface to the First Edition**

### **1 Stata basics**

- 1.1 [Interactive use](#)
- 1.2 [Documentation](#)
- 1.3 [Command syntax and operators](#)
- 1.4 [Do-files and log files](#)
- 1.5 [Scalars and matrices](#)
- 1.6 [Using results from Stata commands](#)
- 1.7 [Global and local macros](#)
- 1.8 [Looping commands](#)
- 1.9 [Mata and Python in Stata](#)
- 1.10 [Some useful commands](#)
- 1.11 [Template do-file](#)
- 1.12 [Community-contributed commands](#)
- 1.13 [Additional resources](#)
- 1.14 [Exercises](#)

### **2 Data management and graphics**

- 2.1 [Introduction](#)
- 2.2 [Types of data](#)
- 2.3 [Inputting data](#)
- 2.4 [Data management](#)
- 2.5 [Manipulating datasets](#)
- 2.6 [Graphical display of data](#)
- 2.7 [Additional resources](#)
- 2.8 [Exercises](#)

### **3 Linear regression basics**

- 3.1 [Introduction](#)
- 3.2 [Data and data summary](#)
- 3.3 [Transformation of data before regression](#)

- 3.4 [Linear regression](#)
- 3.5 [Basic regression analysis](#)
- 3.6 [Specification analysis](#)
- 3.7 [Specification tests](#)
- 3.8 [Sampling weights](#)
- 3.9 [OLS using Mata](#)
- 3.10 [Additional resources](#)
- 3.11 [Exercises](#)

## **[4 Linear regression extensions](#)**

- 4.1 [Introduction](#)
- 4.2 [In-sample prediction](#)
- 4.3 [Out-of-sample prediction](#)
- 4.4 [Predictive margins](#)
- 4.5 [Marginal effects](#)
- 4.6 [Regression decomposition analysis](#)
- 4.7 [Shapley decomposition of relative regressor importance](#)
- 4.8 [Difference-in-differences estimators](#)
- 4.9 [Additional resources](#)
- 4.10 [Exercises](#)

## **[5 Simulation](#)**

- 5.1 [Introduction](#)
- 5.2 [Pseudorandom-number generators](#)
- 5.3 [Distribution of the sample mean](#)
- 5.4 [Pseudorandom-number generators: Further details](#)
- 5.5 [Computing integrals](#)
- 5.6 [Simulation for regression: Introduction](#)
- 5.7 [Additional resources](#)
- 5.8 [Exercises](#)

## **[6 Linear regression with correlated errors](#)**

- 6.1 [Introduction](#)
- 6.2 [Generalized least-squares and FGLS regression](#)
- 6.3 [Modeling heteroskedastic data](#)
- 6.4 [OLS for clustered data](#)
- 6.5 [FGLS estimators for clustered data](#)
- 6.6 [Fixed-effects estimator for clustered data](#)

- 6.7 [Linear mixed models for clustered data](#)
- 6.8 [Systems of linear regressions](#)
- 6.9 [Survey data: Weighting, clustering, and stratification](#)
- 6.10 [Additional resources](#)
- 6.11 [Exercises](#)

## **[7 Linear instrumental-variables regression](#)**

- 7.1 [Introduction](#)
- 7.2 [Simultaneous equations model](#)
- 7.3 [Instrumental-variables regression](#)
- 7.4 [Instrumental-variables example](#)
- 7.5 [Weak instruments](#)
- 7.6 [Diagnostics and tests for weak instruments](#)
- 7.7 [Inference with weak instruments](#)
- 7.8 [Finite sample inference with weak instruments](#)
- 7.9 [Other estimators](#)
- 7.10 [Three-stage least-squares systems estimation](#)
- 7.11 [Additional resources](#)
- 7.12 [Exercises](#)

## **[8 Linear panel-data models: Basics](#)**

- 8.1 [Introduction](#)
- 8.2 [Panel-data methods overview](#)
- 8.3 [Summary of panel data](#)
- 8.4 [Pooled or population-averaged estimators](#)
- 8.5 [Fixed-effects or within estimator](#)
- 8.6 [Between estimator](#)
- 8.7 [Random-effects estimator](#)
- 8.8 [Comparison of estimators](#)
- 8.9 [First-difference estimator](#)
- 8.10 [Panel-data management](#)
- 8.11 [Additional resources](#)
- 8.12 [Exercises](#)

## **[9 Linear panel-data models: Extensions](#)**

- 9.1 [Introduction](#)
- 9.2 [Panel instrumental-variables estimation](#)
- 9.3 [Hausman–Taylor estimator](#)

9.4 [Arellano–Bond estimator](#)

9.5 [Long panels](#)

9.6 [Additional resources](#)

9.7 [Exercises](#)

## 10 [Introduction to nonlinear regression](#)

10.1 [Introduction](#)

10.2 [Binary outcome models](#)

10.3 [Probit model](#)

10.4 [MEs and coefficient interpretation](#)

10.5 [Logit model](#)

10.6 [Nonlinear least squares](#)

10.7 [Other nonlinear estimators](#)

10.8 [Additional resources](#)

10.9 [Exercises](#)

## 11 [Tests of hypotheses and model specification](#)

11.1 [Introduction](#)

11.2 [Critical values and p-values](#)

11.3 [Wald tests and confidence intervals](#)

11.4 [Likelihood-ratio tests](#)

11.5 [Lagrange multiplier test \(or score test\)](#)

11.6 [Multiple testing](#)

11.7 [Test size and power](#)

11.8 [The power onemean command for multiple regression](#)

11.9 [Specification tests](#)

11.10 [Permutation tests and randomization tests](#)

11.11 [Additional resources](#)

11.12 [Exercises](#)

## 12 [Bootstrap methods](#)

12.1 [Introduction](#)

12.2 [Bootstrap methods](#)

12.3 [Bootstrap pairs using the vce\(bootstrap\) option](#)

12.4 [Bootstrap pairs using the bootstrap command](#)

12.5 [Percentile-t bootstraps with asymptotic refinement](#)

12.6 [Wild bootstrap with asymptotic refinement](#)

12.7 [Bootstrap pairs using bsample and simulate](#)

- 12.8 [Alternative resampling schemes](#)
- 12.9 [The jackknife](#)
- 12.10 [Additional resources](#)
- 12.11 [Exercises](#)

## **[13 Nonlinear regression methods](#)**

- 13.1 [Introduction](#)
- 13.2 [Nonlinear example: Doctor visits](#)
- 13.3 [Nonlinear regression methods](#)
- 13.4 [Different estimates of the VCE](#)
- 13.5 [Prediction](#)
- 13.6 [Predictive margins](#)
- 13.7 [Marginal effects](#)
- 13.8 [Model diagnostics](#)
- 13.9 [Clustered data](#)
- 13.10 [Additional resources](#)
- 13.11 [Exercises](#)

## **[14 Flexible regression: Finite mixtures and nonparametric](#)**

- 14.1 [Introduction](#)
- 14.2 [Models based on finite mixtures](#)
- 14.3 [FMM example: Earnings of doctors](#)
- 14.4 [Global polynomials](#)
- 14.5 [Regression splines](#)
- 14.6 [Nonparametric regression](#)
- 14.7 [Partially parametric regression](#)
- 14.8 [Additional resources](#)
- 14.9 [Exercises](#)

## **[15 Quantile regression](#)**

- 15.1 [Introduction](#)
- 15.2 [Conditional quantile regression](#)
- 15.3 [CQR for medical expenditures data](#)
- 15.4 [CQR for generated heteroskedastic data](#)
- 15.5 [Quantile treatment effects for a binary treatment](#)
- 15.6 [Additional resources](#)
- 15.7 [Exercises](#)

## **A Programming in Stata**

- A.1 [Stata matrix commands](#)
- A.2 [Programs](#)
- A.3 [Program debugging](#)
- A.4 [Additional resources](#)

## **B Mata**

- B.1 [How to run Mata](#)
- B.2 [Mata matrix commands](#)
- B.3 [Programming in Mata](#)
- B.4 [Additional resources](#)

## **C Optimization in Mata**

- C.1 [Mata moptimize\(\) function](#)
- C.2 [Mata optimize\(\) function](#)
- C.3 [Additional resources](#)

## **Glossary of abbreviations**

## **References**

## **Author index**

## **Subject index**

# Tables

[2.1 Stata's numeric storage types](#)

[3.1 Postestimation commands](#)

[4.1 DID computation example](#)

[6.1 OLS and FGLS estimators and their estimated variance](#)

[7.1 IV estimators and their asymptotic variances](#)

[8.1 Summary of xt commands for linear panel models](#)

[13.1 Some estimation commands for linear and nonlinear cross-sectional models](#)

# Figures

[1.1 Basic help contents](#)

[2.1 A basic scatterplot of log earnings on hours](#)

[2.2 A more elaborate scatterplot of log earnings on hours](#)

[2.3 Box-and-whisker plots of annual hours for four categories of educational attainment](#)

[2.4 A histogram for log earnings](#)

[2.5 The estimated density of log earnings](#)

[2.6 Histogram and kernel density plot for log earnings](#)

[2.7 Twoway scatterplot and fitted quadratic with confidence bands](#)

[2.8 Local constant plot of log earnings against hours](#)

[2.9 Local linear and lowess plots of log earnings against hours](#)

[2.10 Multiple scatterplots of several variables for each level of education](#)

[3.1 Comparison of densities of level and natural logarithm of medical expenditures](#)

[3.2 Residuals plotted against fitted values after OLS regression](#)

[3.3 Q–Q plot of residuals against normal and kernel density estimate](#)

[4.1 Predictive margins 1\) by age and 2\) by age and gender](#)

[4.2 Predictive margins 1\) by age for men and 2\) by age for women](#)

[5.1  \$\chi^2\(10\)\$  and Poisson\(5\) draws](#)

[5.2 Histogram for one sample of size 30](#)

[5.3 Histogram of the 10,000 sample means, each from a sample of size 30](#)

[5.4 Halton sequence compared with uniform draws](#)

[5.5  \$t\$  statistic density compared with theoretical  \$t\(148\)\$](#)

[6.1 Absolute residuals graphed against  \$x\_2\$  and  \$x\_3\$](#)

[8.1 Time-series plots of log wage against year and weeks worked against year for each of the first 20 observations](#)

[8.2 Overall scatterplot of log wage against experience using all observations](#)

[8.3 Within scatterplot of log-wage deviations from individual means against experience deviations from individual means](#)

[10.1 Probit model and associated ME](#)

[11.1  \$\chi^2\(5\)\$  density compared with 5 times  \$F\(5, 30\)\$  density](#)

[11.2 Power curve for the test of  \$H\_0: \beta\_2 = 2\$  against  \$H\_a: \beta\_2 \neq 2\$  when  \$\beta\_2\$](#)

takes on the values  $\beta_2^{Ha} = 1.6, \dots, 2.4$  under  $H_a$  and  $N = 150$  and  $S = 1000$

11.3 Power curves at test size 0.05 and at sizes 0.05 and 0.10

12.1 Distribution of  $t^*$  from pairs percentile- $t$  bootstrap

14.1 Finite mixture compared with weighted sum of normal random variables

14.2 Finite mixture density of three normal distributions

14.3 Component-wise density of fitted values

14.4 Plot of fitted quadratic and quartic models

14.5 Piecewise linear: Single regressor  $z$  without and with additional regressors

14.6 Natural cubic spline and smoothing spline

14.7 Local linear regression using `lpoly` and `npregress kernel`

15.1 Quantiles of the dependent variable

15.2 QR and OLS coefficients and confidence intervals for each regressor as  $q$  varies from 0 to 1

15.3 Density of  $u$ , quantiles of  $y$ , and scatterplots of  $(y, x_2)$  and  $(y, x_3)$

15.4 Quantile and density plots of log expenditure for those with and without supplementary insurance

# Preface to the Second Edition

*Microeometrics Using Stata*, published in December 2008, was written for Stata 10.1. *Microeconomics Using Stata, Revised Edition*, published in January 2010, was written for Stata 11.0. This second edition is written for Stata 17.

Whereas the scope and coverage of the preceding editions were reasonably synchronized with our own *Microeconomics: Methods and Applications* (Cambridge, 2005), this second edition has broader scope in several respects. We have attempted not only to update our previous coverage to bring it in line with newer tools in the latest edition of Stata but also to bring into the book many topics and methods that are now actively studied and increasingly used in applied microeconomics. This coverage includes several topics, listed below, that were not covered in our 2005 text.

This second edition covers over ten years of both enhancements to Stata and developments in the methods most commonly used in empirical microeconomics analysis. The focus of the book remains the use of linear and nonlinear regression methods for cross-sectional and short panel data. In particular, we give only short treatment to other features of Stata that are useful for data analysis such as data management, use within Stata of other programming languages such as Python, and automated document preparation. The new edition is much expanded and is split into two volumes.

The first volume, comprising chapters 1–15 and Stata and Mata appendixes, focuses on the linear regression model and provides a brief introduction to nonlinear regression models. This volume is an expanded version of chapters 1–10, 12–13, and the appendixes of the first and revised editions. In places, there is greater explanation of underlying methods, and much of the first volume is intended to be suitable for an advanced undergraduate course in addition to serving graduate students and researchers.

The second volume, comprising chapters 16–30, covers the standard nonlinear models as well as more advanced and more recent material. In addition to updated versions of chapters 14–18 of the first edition and the revised edition, the second volume includes new chapters on duration models, treatment effects in randomized control trials, treatment effects with endogenous treatments, parametric models for endogeneity and heterogeneity, spatial regression, semiparametric regression, machine learning and prediction, and Bayesian methods.

Some methods we cover are well established. Other methods we present are in areas of active research, so they may become replaced by better methods. In particular, many methods for causal analysis using observational or experimental methods are still being established and improved upon, at a remarkably rapid pace. This includes inference for instrumental variables with weak instruments, cluster–robust inference with few clusters, treatment-effects estimation with heterogeneous treatment effects, regression discontinuity design, and causal analysis using machine learning methods. Accordingly, we plan to periodically add some supplementary material on the book’s website (<http://cameron.econ.ucdavis.edu/mus2>).

Our target user base consists of practitioners of applied microeconomics. This group is quite diverse in terms of familiarity with the available econometric tools. In deference to such diversity, we have chosen to separate the more advanced aspects of many topics and place them in different parts of the book. This is a challenging task because often the same material could, and in some cases should, appear in several alternative places. To assist the reader, we have provided numerous cross-references and a much lengthier subject index. The reader will benefit from checking out these connections.

Datasets and the do-files used in this book are available on the Stata Press website at <https://www.stata-press.com/data/mus2.html>. Any corrections to the book will be documented at <https://www.stata-press.com/books/microeconomics-stata/>.

The preparation of this second edition has benefited from generous help from many sources. We thank our colleagues, coauthors, students, and many users of the previous editions for their suggested improvements, for reading

parts of the book, for permission to use datasets developed in joint research, and for encouragement to proceed with the project. We have benefited from presenting some of the material in various short courses around the world and from positive feedback from readers of the earlier editions that encouraged writing this updated edition. Colin Cameron would especially like to thank Shu Shen, Takuya Ura, Oscar Jorda, Marianne Bitler, the broader econometrics and empirical microeconomics community at the University of California–Davis, and Doug Miller and Adrian Pagan. Pravin Trivedi gratefully acknowledges the support provided by the School of Economics, University of Queensland. We thank Yulia Marchenko and Nikolay Balov for very detailed comments on the Bayesian chapters, and Kristin MacDonald for a careful reading of the final draft of the book. We thank David Culwell for his excellent editing and Stephanie White for managing the LaTeX formatting and production of this book. Most especially, both authors acknowledge their debt of gratitude to David Drukker for extensive feedback on many aspects of the material in the book throughout this project, including a complete reading, as well as feedback on the substantive aspects of applying the econometric and statistical tools. Finally, we thank our respective families for their patience and understanding during the long gestation period of the evolution of this project.

Davis, CA  
Charlottesville, VA  
June 2022

A. Colin Cameron  
Pravin K. Trivedi

# Preface to the First Edition

This book explains how an econometrics computer package, Stata, can be used to perform regression analysis of cross-section and panel data. The term microeconomics is used in the book title because the applications are to economics-related data and because the coverage includes methods such as instrumental-variables regression that are emphasized more in economics than in some other areas of applied statistics. However, many issues, models, and methodologies discussed in this book are also relevant to other social sciences.

The main audience is graduate students and researchers. For them, this book can be used as an adjunct to our own *Microeometrics: Methods and Applications* ([Cameron and Trivedi 2005](#)), as well as to other graduate-level texts such as [Greene \(2008\)](#) and [Wooldridge \(2010\)](#). By comparison to these books, we present little theory and instead emphasize practical aspects of implementation using Stata. More advanced topics we cover include quantile regression, weak instruments, nonlinear optimization, bootstrap methods, nonlinear panel-data methods, and Stata's matrix programming language, Mata.

At the same time, the book provides introductions to topics such as ordinary least-squares regression, instrumental-variables estimation, and logit and probit models so that it is suitable for use in an undergraduate econometrics class, as a complement to an appropriate undergraduate-level text. The following table suggests sections of the book for an introductory class, with the caveat that in places formulas are provided using matrix algebra.

Stata basics	Chapter 1.1–1.4
Data management	Chapter 2.1–2.4, 2.6
OLS	Chapter 3.1–3.6
Simulation	Chapter 4.6–4.7
Generalized least squares (heteroskedasticity)	Chapter 5.3
Instrumental variables	Chapter 6.2–6.3
Linear panel data	Chapter 8
Logit and probit models	Chapter 14.1–14.4
Tobit model	Chapter 16.1–16.3

Although we provide considerable detail on Stata, the treatment is by no means complete. In particular, we introduce various Stata commands but avoid detailed listing and description of commands as they are already well documented in the Stata manuals and online help. Typically, we provide a pointer and a brief discussion and often an example.

As much as possible, we provide template code that can be adapted to other problems. Keep in mind that to shorten output for this book, our examples use many fewer regressors than necessary for serious research. Our code often suppresses intermediate output that is important in actual research, because of extensive use of command `quietly` and options `nolog`, `nodots`, and `noheader`. And we minimize the use of graphs compared with typical use in exploratory data analysis.

We have used Stata 10, including Stata updates.<sup>1</sup> Instructions on how to obtain the datasets and the do-files used in this book are available on the Stata Press website at <https://www.stata-press.com/data/mus.html>. Any corrections to the book will be documented at <https://www.stata-press.com/books/mus.html>.

We have learned a lot of econometrics, in addition to learning Stata, during this project. Indeed, we feel strongly that an effective learning tool for econometrics is hands-on learning by opening a Stata dataset and seeing the effect of using different methods and variations on the methods, such as using robust standard errors rather than default standard errors. This method is beneficial at all levels of ability in econometrics. Indeed, an efficient way

of familiarizing yourself with Stata's leading features might be to execute the commands in a relevant chapter on your own dataset.

We thank the many people who have assisted us in preparing this book. The project grew out of our 2005 book, and we thank Scott Parris for his expert handling of that book. Juan Du, Qian Li, and Abhijit Ramalingam carefully read many of the book chapters. Discussions with John Daniels, Oscar Jorda, Guido Kuersteiner, and Doug Miller were particularly helpful. We thank Deirdre Patterson for her excellent editing and Lisa Gilmore for managing the LaTeX formatting and production of this book. Most especially, we thank David Drukker for his extensive input and encouragement at all stages of this project, including a thorough reading and critique of the final draft, which led to many improvements in both the econometrics and Stata components of this book. Finally, we thank our respective families for making the inevitable sacrifices as we worked to bring this multiyear project to completion.

Davis, CA  
Bloomington, IN  
October 2008

A. Colin Cameron  
Pravin K. Trivedi

---

<sup>1</sup> To see whether you have the latest update, type `update query`. For those with earlier versions of Stata, some key changes are the following: Stata 9 introduced the matrix programming language, Mata. The syntax for Stata 10 uses the `vce(robust)` option rather than the `robust` option to obtain robust standard errors. A mid-2008 update of version 10 introduced new random-number functions, such as `runiform()` and `rnormal()`.



# Chapter 1

## Stata basics

This chapter provides some of the basic information about issuing commands in Stata. Sections [1.1–1.3](#) enable a first-time user to begin using Stata interactively. In this book, we instead emphasize storing these commands in a text file, called a Stata do-file, that is then executed. This is presented in section [1.4](#). Sections [1.5–1.8](#) present more advanced Stata material that might be skipped on a first reading.

The chapter concludes with a summary of some commonly used Stata commands and with a template do-file that demonstrates many of the tools introduced in this chapter. Chapters [2](#) and [3](#) then demonstrate many of the Stata commands and tools used in applied microeconomics. Additional features of Stata are introduced throughout the book and the appendixes.

## 1.1 Interactive use

Interactive use means that Stata commands are initiated from within Stata.

A graphical user interface (GUI) for Stata is available. This enables almost all Stata commands to be selected from drop-down menus. Interactive use is then especially easy because there is no need to know in advance the Stata command.

All implementations of Stata allow commands to be directly typed in; for example, entering `summarize` yields summary statistics for the current dataset. This is the primary way that Stata is used because it is considerably faster than working through drop-down menus. Furthermore, for most analyses, the standard procedure is to aggregate the various commands needed into one file called a do-file (see section [1.4](#)) that can be run with or without interactive use. We therefore provide little detail on the Stata GUI.

For new Stata users, we suggest entering Stata, usually by clicking on the Stata icon, opening one of the Stata example datasets, and doing some basic statistical analysis. To obtain example data, select **File > Example datasets...**, meaning from the **File** menu, select the entry **Example datasets....**. Then, click on the link to **Example datasets installed with Stata**. Work with `auto.dta`; this is used in many of the introductory examples presented in the Stata documentation. First, select **describe** to obtain descriptions of the variables in the dataset. Second, select **use** to read the dataset into Stata. You can then obtain summary statistics either by typing `summarize` in the Command window or by selecting **Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Summary statistics**. You can run a simple regression by typing `regress mpg weight` or by selecting **Statistics > Linear models and related > Linear regression** and then using the drop-down lists in the **Model** tab to choose `mpg` as the dependent variable and `weight` as the independent variable.

The Stata manual [GSM] *Getting Started with Stata for Mac*, or its Unix and Windows versions [GSU] *Getting Started with Stata for Unix* and [GSW]

*Getting Started with Stata for Windows*, is very helpful, especially [GS] 1 Introducing Stata—sample session, which uses typed-in commands, and [GS] 2 The Stata user interface.

The extent to which you use Stata in interactive mode is really a personal preference. There are several reasons for at least occasionally using interactive mode. First, it can be useful for learning how to use Stata. Second, it can be useful for exploratory analysis of datasets because you can see in real time the effect of, for example, adding or dropping regressors. If you do this, however, be sure to first start a session log file (see section 1.4) that saves the commands and resulting output. Third, you can use `help` and related commands to obtain online information about Stata commands. Fourth, one way to implement the preferred method of running do-files is to use the Stata Do-file Editor in interactive mode.

Finally, components of a given version of Stata, such as version 17, are periodically updated. Entering `update` determines the current update level and provides the option to install official updates to Stata. You can also install community-contributed commands in interactive mode once the relevant software is located by using, for example, the `search` command.

## 1.2 Documentation

Stata documentation is extensive; you can find it in Stata (online) or on the web.

### 1.2.1 Stata manuals

For first-time users, see [GSM] *Getting Started with Stata for Mac* or [GSU] *Getting Started with Stata for Unix* or [GSW] *Getting Started with Stata for Windows*. The most useful manual is [U] *User’s Guide*. Entries within manuals are referred to using shorthand such as [U] **11.1.4 in range**, which denotes section 11.1.4 of [U] *User’s Guide* on the topic **in range**.

Many commands are described in [R] *Base Reference Manual*. Not all Stata commands appear here, however, because some appear instead in the appropriate topical reference manual. These topical reference manuals are [BAYES] *Bayesian Analysis*, [CM] *Choice Models*, [D] *Data Management*, [DSGE] *Dynamic Stochastic General Equilibrium Models*, [ERM] *Extended Regression Models*, [FMM] *Finite Mixture Models*, [FN] *Functions*, [G] *Graphics*, [IRT] *Item Response Theory*, [LASSO] *Lasso*, [M] *Mata*, [META] *Meta-Analysis*, [ME] *Multilevel Mixed Effects*, [MI] *Multiple-Imputation*, [MV] *Multivariate Statistics*, [P] *Programming*, [PSS] *Power, Precision, and Sample Size*, [RPT] *Reporting*, [SEM] *Structural Equation Modeling*, [SP] *Spatial Autoregressive Models*, [ST] *Survival Analysis*, [SVY] *Survey Data*, [TE] *Treatment-Effects*, [TABLES] *Customizable Tables and Collected Results*, [TS] *Time-Series*, and [XT] *Longitudinal-Data/Panel-Data*. For example, the `generate` command appears in [D] **generate** rather than in [R].

For a complete list of documentation, see [U] **1 Read this—it will help** and also [I] *Index*.

### 1.2.2 Additional Stata resources

The *Stata Journal* (SJ) and its predecessor, the *Stata Technical Bulletin* (STB), present examples and code that go beyond the current installation of Stata. SJ articles over three years old and all STB articles are available online

from the Stata website at no charge. You can find this material by using various Stata help commands given later in this section, and you can often install code as a free community-contributed command.

The Stata website has a lot of information. This includes a summary of what Stata does. A good place to begin is <https://www.stata.com/support/>. In particular, see the answers to frequently asked questions (FAQ) and video tutorials. The blog posts at <https://blog.stata.com> include many useful detailed Stata examples.

The University of California–Los Angeles website <https://stats.oarc.ucla.edu/stata/> provides many introductory tutorials.

### 1.2.3 The `help` command

Stata has extensive help available once you are in the program.

The `help` command is most useful if you already know the name of the command for which you need help. For example, for help on the `regress` command, type

```
. help regress  
(output omitted)
```

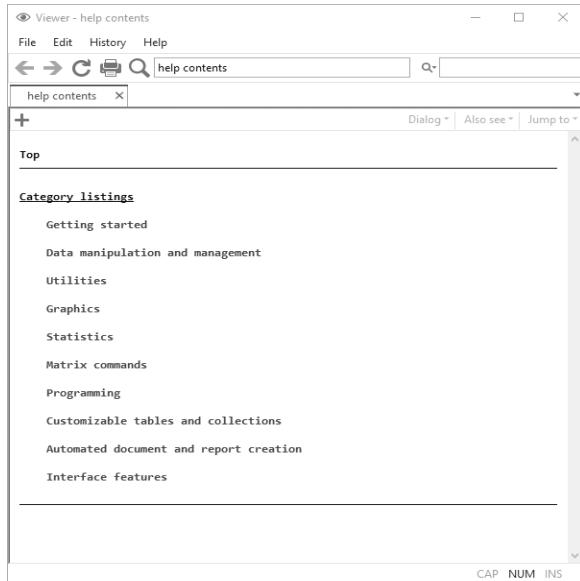
Note that here and elsewhere the dot (.) is not typed in but is provided to enable distinction between Stata commands (preceded by a dot) and subsequent Stata output, which appears with no dot.

The `help` command is also useful if you know the class of commands for which you need help. For example, for help on functions, type

```
. help function  
(output omitted)
```

Often, however, you need to start with the basic `help contents` command, which will open the Viewer window shown in figure 1.1.

```
. help contents
```



**Figure 1.1.** Basic help contents

For further details, click on a category and subsequent subcategories.

For help with the Stata matrix programming language, Mata, add the term `mata` after `help`. Often, for Mata, it is necessary to start with the very broad command

```
. help mata  
(output omitted)
```

and then narrow the results by selecting the appropriate categories and subcategories.

#### 1.2.4 The search and net search commands

The `search` command does a keyword database and Internet search. It is especially useful if you do not know the Stata command name or if you want to find the many places that a command or method might be used. The default for `search` is to obtain all available information from official Stata help files, books, blogs, FAQ, examples, the SJ, and the STB, and Stata code from some Internet sources, notably the Statistical Software Components Archive maintained by the Boston College Department of Economics. For example, for ordinary least squares (OLS), the command

```
. search ols  
(output omitted)
```

finds references in the manuals [R], [BAYES], [MV], and [SVY]; in books; in blogs; in FAQ; in examples; in the SJ and the STB; and finds many packages on the web. It also gives `help` commands that you can click on to get further information without the need to consult the manuals.

The default of the `search` command is to find entries that match all the keywords provided. To instead find occurrence of any of the keywords provided, use the `or` option. For example, typing

```
. search weak instr, or  
(output omitted)
```

finds joint occurrences of any words beginning with the letters “weak” and the letters “instr”.

The `search` command has several other options, including the `net` option, which searches only the Internet for installable packages, including code from the SJ, the STB and the Statistical Software Components Archive.

## 1.3 Command syntax and operators

Stata command syntax describes the rules of the Stata programming language.

### 1.3.1 Basic command syntax

The basic command syntax is almost always some subset of

```
[prefix:] command [varlist] [= exp] [if] [in] [weight] [using filename]  
[ , options ]
```

The brackets denote qualifiers that in most instances are optional. Words in the typewriter font are to be typed into Stata as they appear on the page. Italicized words are to be substituted by the user, where

- *prefix* denotes a command that repeats execution of *command* or modifies the input or output of *command* with,
- *command* denotes a Stata command,
- *varlist* denotes a list of variable names,
- *exp* is a mathematical expression,
- *if* identifies observations via an expression
- *in* denotes a range of observations,
- *weight* denotes a weighting expression,
- *filename* is a filename, and
- *options* denotes one or more options that apply to *command*.

The greatest variation across commands is in the available options. Commands can have many options, and these options can also have options, which are given in parentheses.

Stata is case sensitive. We generally use lowercase throughout, though occasionally we use uppercase for model names.

Commands and output are displayed following the style for Stata manuals. For Stata commands given in the text, the typewriter font is used. For example, for OLS, we use the `regress` command. For displayed

commands and output, the commands have the prefix `.` (a period followed by a space), whereas output has no prefix. For Mata commands, the displayed prefix is a colon `(:)` rather than a period. Output from commands that span more than one line has the continuation prefix `>` (greater-than sign). For a Stata or Mata program, the lines within the program do not have a prefix.

### 1.3.2 Example: The summarize command

The `summarize` command provides descriptive statistics (for example, mean, standard deviation) for one or more variables.

You can obtain the syntax of `summarize` by typing `help summarize`. This yields output including

```
summarize [ varlist ] [ if ] [ in ] [ weight ] [ , options ]
```

It follows that, at the minimum, we can give the command without any qualifiers. Unlike some commands, `summarize` does not use `[= exp]` or `[using filename]`.

As an example, we use a commonly used, illustrative dataset installed with Stata called `auto.dta`, which has information on various attributes of 74 automobiles. You can read this dataset into memory by using the `sysuse` command, which accesses Stata-installed datasets. To read in the data and obtain descriptive statistics, we type

```
. * Use a dataset that was installed with Stata
. sysuse auto
(1978 automobile data)
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
make	0				
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
rep78	69	3.405797	.9899323	1	5
headroom	74	2.993243	.8459948	1.5	5
trunk	74	13.75676	4.277404	5	23
weight	74	3019.459	777.1936	1760	4840
length	74	187.9324	22.26634	142	233
turn	74	39.64865	4.399354	31	51
displacement	74	197.2973	91.83722	79	425
gear_ratio	74	3.014865	.4562871	2.19	3.89
foreign	74	.2972973	.4601885	0	1

The dataset comprises 12 variables for 74 automobiles. The average price of the automobiles is \$6,165, and the standard deviation is \$2,949. The column `obs` gives the number of observations for which data are available for each variable. The `make` variable has 0 observations because it is a string (or text) variable giving the make of the automobile, and summary statistics are not applicable to a nonnumeric variable. The `rep78` variable is available for only 69 of the 74 observations.

A more focused use of `summarize` restricts attention to selected variables and uses one or more of the available options. For example,

```
. * Summary statistics of selected variables
. summarize mpg price weight, separator(1)
```

Variable	Obs	Mean	Std. dev.	Min	Max
mpg	74	21.2973	5.785503	12	41
price	74	6165.257	2949.496	3291	15906
weight	74	3019.459	777.1936	1760	4840

provides descriptive statistics for the `mpg`, `price`, and `weight` variables. The option `separator(1)` inserts a line between the output for each variable.

### 1.3.3 Example: The regress command

The `regress` command implements OLS regression.

You can obtain the syntax of `regress` by typing `help regress`. This yields output including

```
regress depvar [indepvars] [if] [in] [weight] [, options]
```

It follows that, at the minimum, we need to include the variable name for the dependent variable (in that case, the regression is on an intercept only). Although not explicitly stated, prefixes can be used. Many estimation commands have similar syntax.

Suppose that we want to run an OLS regression of the `mpg` variable (fuel economy in miles per gallon) on `price` (auto price in dollars) and `weight` (weight in pounds). The basic command is simply

.	*	OLS regression
.		<code>regress mpg price weight</code>
<hr/>		
Source	SS	df MS
Model	1595.93249	2 797.966246
Residual	847.526967	71 11.9369995
Total	2443.45946	73 33.4720474
<hr/>		
mpg	Coefficient	Std. err. t P> t  [95% conf. interval]
price	-.0000935	.0001627 -0.57 0.567 -.000418 .0002309
weight	-.0058175	.0006175 -9.42 0.000 -.0070489 -.0045862
_cons	39.43966	1.621563 24.32 0.000 36.20635 42.67296

The coefficient of  $-0.0058175$  for `weight` implies that fuel economy falls by 5.8 miles per gallon when the car's weight increases by 1,000 pounds.

A more complicated version of `regress` that demonstrates much of the command syntax is the following:

```
. by foreign: regress mpg price weight if weight < 4000, vce(robust)  
(output omitted)
```

For each value of the `foreign` variable, here either 0 or 1, this command fits distinct OLS regressions of `mpg` on `price` and `weight`. The `if` qualifier limits the sample to cars with `weight` less than 4,000 pounds. The `vce(robust)` option leads to heteroskedasticity-robust standard errors being used.

The `by` prefix is an example of a command prefix that repeats execution of the subsequent command and is usually followed by a colon. The command `help prefix` lists the available command prefixes, including `by`, `bysort`, `bayes`, `bootstrap`, `simulate`, `svy`, and `quietly`.

Output from commands is not always desired. We can suppress output by using the `quietly` prefix. For example,

```
. * Suppress output from command  
. quietly regress mpg price weight
```

The `quietly` prefix does not require a colon, for historical reasons, even though it is a command prefix. In this book, we use this prefix extensively to suppress extraneous output, abbreviated to `qui` to enable more commands to fit in one line.

The preceding examples used one of the available options for `regress`. From `help regress`, we find that the `regress` command has the following options: `noconstant`, `hascons`, `tsscons`, `vce(vcetype)`, `level(#)`, `beta`, `eform(string)`, `depname(varname)`, *display\_options*, `noheader`, `notable`, `plus`, `mse1`, and `coeflegend`.

### 1.3.4 Factor variables

Factor variables enable reference to a set of indicator variables based on a (nonnegative and integer-valued) categorical variable by inserting the `i.` operator in front of the name of the categorical variable. Factor variables can be used in the variable list of most Stata commands.

As an example, consider the variable `rep78`, the repair record in 1978. This takes five distinct values that are 1, 2, 3, 4, and 5, though any other nonnegative integer values will do. Additionally, variable `rep78` is missing for five observations. We have

```
. * Factor variables for rep78
. summarize i.rep78
```

Variable	Obs	Mean	Std. dev.	Min	Max
rep78					
1	69	.0289855	.1689948	0	1
2	69	.115942	.3225009	0	1
3	69	.4347826	.4993602	0	1
4	69	.2608696	.4423259	0	1
5	69	.1594203	.3687494	0	1

We can also include factor variables in regression commands.

```
. * Factor variables for rep78 in regression
. regress mpg i.rep78
```

Source	SS	df	MS	Number of obs	=	69
Model	549.415777	4	137.353944	F(4, 64)	=	4.91
Residual	1790.78712	64	27.9810488	Prob > F	=	0.0016
Total	2340.2029	68	34.4147485	R-squared	=	0.2348
				Adj R-squared	=	0.1869
				Root MSE	=	5.2897

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]
rep78					
2	-1.875	4.181884	-0.45	0.655	-10.22927 6.479274
3	-1.566667	3.863059	-0.41	0.686	-9.284014 6.150681
4	.6666667	3.942718	0.17	0.866	-7.209818 8.543152
5	6.363636	4.066234	1.56	0.123	-1.759599 14.48687
_cons	21	3.740391	5.61	0.000	13.52771 28.47229

The default with regression commands is to omit one category, that for the lowest value taken by the categorical variable. For variable `rep78`, this is the value 1. To see what category is the base (or omitted) category, add the `allbaselevels` option after the command (here `regress`). To change the base category, use the `ib.` operator instead of the `i.` operator. For example, the `regress mpg ib2.rep78` command will omit the category `rep78 = 2`, and the `regress mpg ib(last).rep78` command will omit the highest-valued category (here `rep78 = 5`). Alternatively, the command `fvset base 5 rep78` will permanently set the fifth category to be the base category.

A complete set of indicators, with no category omitted, is included using the `ibn.` operator with the `hascons` option, which omits the intercept. For

example,

```
. * Factor variables for rep78 - no category is omitted  
. regress mpg ibn.rep78, hascons
```

Source	SS	df	MS	Number of obs	=	69
Model	549.415777	4	137.353944	F(4, 64)	=	4.91
Residual	1790.78712	64	27.9810488	Prob > F	=	0.0016
Total	2340.2029	68	34.4147485	R-squared	=	0.2348
				Adj R-squared	=	0.1869
				Root MSE	=	5.2897
mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
rep78						
1	21	3.740391	5.61	0.000	13.52771	28.47229
2	19.125	1.870195	10.23	0.000	15.38886	22.86114
3	19.43333	.9657648	20.12	0.000	17.504	21.36267
4	21.66667	1.246797	17.38	0.000	19.1759	24.15743
5	27.36364	1.594908	17.16	0.000	24.17744	30.54983

A complete set of interactions between two (or more) categorical variables can be created using the # operator. For example, consider an interaction between categorical variable `rep78` and categorical variable `foreign` (a binary indicator). We have

```
. * Factor variables for interaction between two categorical variables
. regress mpg i.rep78##i.foreign, allbaselevels
note: 1b.rep78#1.foreign identifies no observations in the sample.
note: 2.rep78#1.foreign identifies no observations in the sample.
```

Source	SS	df	MS	Number of obs	=	69
Model	839.550121	7	119.935732	F(7, 61)	=	4.88
Residual	1500.65278	61	24.6008652	Prob > F	=	0.0002
Total	2340.2029	68	34.4147485	R-squared	=	0.3588
				Adj R-squared	=	0.2852
				Root MSE	=	4.9599

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]
rep78#foreign					
1#Domestic	0	(base)			
1#Foreign	0	(empty)			
2#Domestic	-1.875	3.921166	-0.48	0.634	-9.715855
2#Foreign	0	(empty)			
3#Domestic	-2	3.634773	-0.55	0.584	-9.268178
3#Foreign	2.333333	4.527772	0.52	0.608	5.268178
4#Domestic	-2.555556	3.877352	-0.66	0.512	-6.720507
4#Foreign	3.888889	3.877352	1.00	0.320	11.38717
5#Domestic	11	4.959926	2.22	0.030	5.19769
5#Foreign	5.333333	3.877352	1.38	0.174	11.64213
_cons	21	3.507197	5.99	0.000	20.91798
					13.98693
					28.01307

Here the base (omitted) category is `rep78 = 1` and `foreign = 0` (the lowest-valued joint category). Additionally, there are zero observations falling into two of the categories: `rep78 = 1` and `foreign = 1`, and `rep78 = 2` and `foreign = 1`.

The `#` operator creates a factorial interaction that includes sets of indicator variables for each of the two categorical variables, in addition to the interactions given by the `#` operator. For example, the command `regress mpg i.rep78##i.foreign` is equivalent to the command `regress mpg i.rep78 i.foreign i.rep78##i.foreign`.

Factor-variable operators can also be used to create interactions between indicator variables and continuous regressors. In that case, the prefix `c.` needs to be used to signal that the interaction is with a continuous variable. For example,

```
. * Factor variables for interaction between categorical and continuous variables
. regress mpg i.rep78#c.weight
```

Source	SS	df	MS	Number of obs	=	69
Model	1535.21253	5	307.042506	F(5, 63)	=	24.03
Residual	804.99037	63	12.7776249	Prob > F	=	0.0000
				R-squared	=	0.6560
				Adj R-squared	=	0.6287
Total	2340.2029	68	34.4147485	Root MSE	=	3.5746
mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
rep78#c.weight						
1	-.0056832	.0010153	-5.60	0.000	-.0077122	-.0036542
2	-.0058149	.0006781	-8.57	0.000	-.00717	-.0044597
3	-.005717	.0005886	-9.71	0.000	-.0068932	-.0045409
4	-.0057904	.0006745	-8.58	0.000	-.0071383	-.0044424
5	-.0051682	.0009273	-5.57	0.000	-.0070212	-.0033151
_cons	38.51076	1.926584	19.99	0.000	34.66078	42.36073

In this continuous interaction example, there is no omitted category—all five possible values of `rep78` are interacted with the continuous variable `weight`.

Factor-variable operators also permit interaction of continuous variables with continuous variables. For example, the following performs OLS regression of `mpg` on `price` and a quadratic in `weight`.

```
. * Factor variables for interaction between two continuous variables
. regress mpg price c.weight c.weight#c.weight, noheader
```

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
price	-.0002597	.0001696	-1.53	0.130	-.000598	.0000786
weight	-.016047	.0040403	-3.97	0.000	-.024105	-.0079889
c.weight#						
c.weight	1.72e-06	6.71e-07	2.56	0.013	3.79e-07	3.06e-06
_cons	54.66807	6.150716	8.89	0.000	42.40086	66.93529

In total, there are five factor-variable operators: `i.`, `c.`, `o.`, `#`, and `##`. The `o.` operator is used to omit a continuous variable (for example, `o.price`) or an indicator variable (for example, `o5.rep78` to omit the indicator variable `rep78=5`).

For more on factor variables, type `help factor variables`, or see [U] 11.4.3 Factor variables and [U] 26 Working with categorical data and factor variables. To check whether the `regress` command, for example, supports factor variables, type the command `help regress` and the output below the syntax summary includes a note that “*indepvars* may contain factor variables; see `fvvarlist`.”

### 1.3.5 Abbreviations, case sensitivity, and wildcards

Commands and parts of commands can be abbreviated to the shortest string of characters that uniquely identify them, often just two or three characters. For example, we can shorten `summarize` to `su`. For expositional clarity, we do not use such abbreviations in this book; two notable exceptions are that we use `qui` rather than `quietly` and that we may use abbreviations in the options to graphics commands because these commands can get very lengthy. Not using abbreviations makes it much easier to read your do-files.

Variable names can be up to 32 characters long, where the characters can be A–Z, a–z, any Unicode letter, 0–9, and \_ (underscore). Some names, such as `in`, are reserved. Stata is case sensitive, and the norm is to use lowercase.

We can use the wildcard \* (asterisk) for variable names in commands, provided there is no ambiguity such as two potential variables for a one-variable command. For example,

```
. * Wildcard (asterisk) example
. summarize t*
```

Variable	Obs	Mean	Std. dev.	Min	Max
trunk	74	13.75676	4.277404	5	23
turn	74	39.64865	4.399354	31	51

provides summary statistics for all variables with names beginning with the letter `t`. Where ambiguity may arise, wildcards are not permitted.

### 1.3.6 Arithmetic, relational, and logical operators

The arithmetic operators in Stata are + (addition), - (subtraction), \* (multiplication), / (division), ^ (raised to a power), and the prefix - (negation). For example, to compute and display  $-2 \times \{9/(8+2-7)\}^2$ , which simplifies to  $-2 \times 3^2$ , we type

```
. * Arithmetic example and display of result  
. display -2*(9/(8+2-7))^2  
-18
```

If the arithmetic operation is not possible, or data are not available to perform the operation, then a missing value denoted by . is displayed. For example,

```
. * Missing value created by impossible arithmetic operation  
. display 2/0  
. .
```

The relational operators are > (greater than), < (less than), >= (greater than or equal), <= (less than or equal), == (equal), and != (not equal). These are the obvious symbols, except that a pair of equal signs is used for equality and != denotes not equal. Relational operators are often used in if qualifiers that define the sample for analysis.

Logical operators return 1 for true and 0 for false. The logical operators are & (and), | (or), and ! (not). The operator ~ can be used in place of !. Logical operators are also used to define the sample for analysis. For example, to restrict regression analysis to smaller less expensive cars, type

```
* Example of logical operators  
. regress mpg price weight if weight <= 4000 & price <= 10000  
(output omitted)
```

The string operator + is used to concatenate two strings into a single, longer string.

The order of evaluation of all operators is ! (or ~), ^, - (negation), /, \*, -, (subtraction), +, != (or ~=), >, <, <=, >=, ==, &, and |.

### 1.3.7 Error messages

Stata produces error messages when a command fails. These messages are brief, but a fuller explanation can be obtained from the manual or directly from Stata.

For example, if we regress mpg on notthere but the notthere variable does not exist, we get

```
. regress mpg notthere  
variable notthere not found  
r(111);
```

Here r(111) denotes return code 111. You can obtain further details by clicking on r(111) or, if in interactive mode, by typing

```
. search rc 111  
(output omitted)
```

## 1.4 Do-files and log files

For Stata analysis requiring many commands, or requiring lengthy commands, it is best to collect all the commands into a program (or script) that is stored in a text file called a do-file.

In this book, we perform data analysis using a do-file. We assume that the do-file and, if relevant, any input and output files are in a common directory and that Stata is executed from that directory. Then, we need to provide only the filename rather than the complete directory structure. For example, we can refer to a file as `mus202data.dta` rather than `C:\mus\chapter2\mus202data.dta`.

### 1.4.1 Writing a do-file

A do-file is a text file with extension `.do` that contains a series of Stata commands.

As an example, we write a two-line program that reads in the Stata example dataset `auto.dta` and then presents summary statistics for the `mpg` variable, which we already know is in the dataset. The commands are `sysuse auto.dta, clear`, where the `clear` option is added to remove the current dataset from memory, and `summarize mpg`. The two commands are to be collected into a command file called a do-file. The filename should include no spaces, and the file extension is `.do`.

In this example, we suppose this file is given the name `example.do` and is stored in the current working directory.

To see the current directory, type `cd` without any arguments. To change to another directory, type `cd` with an argument. For example, in Windows, to change to the directory `C:\Program Files\Stata17\`, we type

```
. cd "c:\Program Files\Stata17"  
c:\Program Files\Stata17
```

The directory name is given in double quotes because it includes spaces. Otherwise, the double quotes are unnecessary.

One way to create the do-file is to start Stata and use the Do-file Editor. Within Stata, we select **Window > Do-file Editor > New Do-file Editor**, type in the commands, and save the do-file.

Alternatively, type in the commands outside Stata by using a preferred text editor. Ideally, this text editor supports multiple windows, reads large files (datasets or output), and gives line numbers and column numbers.

The `type` command lists the contents of the previously created file. We have

```
. type example.do  
sysuse auto, clear  
summarize mpg
```

### 1.4.2 Running do-files

You can run (or execute) an already-written do-file by using the Command window. Start Stata, and, in the Command window, change directory (`cd`) to the directory that has the do-file, and then issue the `do` command. We obtain

```
. do example  
. sysuse auto, clear  
(1978 automobile data)  
. summarize mpg
```

Variable	Obs	Mean	Std. dev.	Min	Max
mpg	74	21.2973	5.785503	12	41

```
.  
end of do-file
```

where we assume that `example.do` is in directory `C:\Program Files\Stata17\`.

An alternative simpler method is to run the do-file from the Do-file Editor. Select **Window > Do-file Editor > New Do-file Editor**, select **File > Open...** and the appropriate file, and finally select **Tools > Execute (do)**.

An advantage to using the Do-file Editor is that you can highlight or select just part of the do-file and then execute this part by selecting **Tools** > **Execute selection (include)**. In either case, you can more simply hit the execute icon (the rightmost icon). This method is especially useful when developing a program because it is easy to make modifications in the Do-file Editor and reexecute.

Finally, and often most simply, to execute `example.do`, for example, double-click on `example.do` in File Explorer. This initiates Stata and opens `example.do` in the Do-file Editor. Furthermore, this sets the working directory to be the directory that `example.do` was in.

### 1.4.3 Log files

By default, Stata output is sent to the screen. For reproducibility, you should save this output in a separate file. Another advantage to saving output is that lengthy output can be difficult to read on the screen; it can be easier to review results by viewing an output file using a text editor.

A Stata output file is called a log file. It stores the commands in addition to the output from these commands. The default extension for a plain text log file is `.log`, but you can choose an alternative extension, such as `.txt`. An extension name change may be worthwhile because several other programs, such as LaTeX compilers, also create files with the `.log` extension. Log files can be read either as standard text or in a special Stata code called `smcl` (Stata Markup and Control Language). We use text throughout this book because it is easier to read in a text editor. A useful convention can be to give the log the same filename as that for the do-file. For instance, for `example.do`, we save the output as `example.txt`.

A log file is created by using the `log` command. In a typical analysis, the do-file will change over time, in which case the output file will also change. The Stata default is to protect against an existing log being accidentally overwritten. To create a log file in text form named `example.txt`, you usually type

```
. * Create log file as a text file  
. log using example.txt, text replace
```

The `replace` option permits the existing version of `example.txt`, if there is one, to be overwritten. Without `replace`, Stata will refuse to open the log file if there is already a file called `example.txt`.

In some cases, we may not want to overwrite the existing log, in which case we would not specify the `replace` option. The most likely reason for preserving a log is that it contains important results, such as those from final analysis. Then it can be good practice to rename the log after analysis is complete. Thus, `example.txt` might be renamed `example07052021.txt`.

When a program is finished, you should close the log file by typing `log close`.

The log can be very lengthy. If you need a hard copy, you can edit the log to include only essential results. The text editor you use should use a monospace font such as Courier New, where each character takes up the same space, so that output table columns will be properly aligned.

The log file includes the Stata commands, with a dot (.) prefix, and the output. You can use a log file to create a do-file, if a do-file does not already exist, by deleting the dot and all lines that are command results (no dot). By this means, you can do initial work using the Stata GUI and generate a do-file from the session, provided that you created a log file at the beginning of the session. The `cmdlog` command creates a file that contains just the typed commands.

#### 1.4.4 A three-step process

Data analysis using Stata can repeatedly use the following three-step process:

1. Create or change the do-file.
2. Execute the do-file in Stata.
3. Read the resulting log with a text editor.

If the Stata Do-file Editor is used, then one can execute highlighted lines of code, rather than the entire do-file.

The initial do-file can be written by editing a previously written do-file that is a useful template or starting point, especially if it uses the same dataset or the same commands as the current analysis. The resulting log may include Stata errors or estimation results that lead to changes in the original do-file and so on.

Suppose we have fit several models and now want to fit an additional model. In interactive mode, we would type in the new command, execute it, and see the results. Using the three-step process, we add the new command to the do-file, execute the do-file or a subcomponent, and read the new output. Because many Stata programs execute in seconds, this adds little extra time compared with using interactive mode, and it has the benefit of having a do-file that can be modified for later use.

#### 1.4.5 Comments and long lines

Stata do-files can include comments. This can greatly increase understanding of a program, which is especially useful if you return to a program and its output a year or two later. Lengthy single-line comments can be allowed to span several lines, ensuring readability. There are several ways to include comments:

- For single-line comments, begin the line with an asterisk (\*); Stata ignores such lines.
- For a comment on the same line as a Stata command, use two slashes (//) after the Stata command.
- For multiple-line comments, place the commented text between slash-star /\*) and star-slash (\*/).

The Stata default is to view each line as a separate Stata command, where a line continues until a carriage return (end-of-line or *Enter* key) is encountered. Some commands, such as those for nicely formatted graphs, can be very long. For readability, these commands need to span more than one line. The easiest way to break a line at, say, the 70th column is by using three slashes (///) and then continuing the command on the next line.

The following do-file code includes several comments to explain the program and demonstrates how to allow a command to span more than one line.

```
* Demonstrate use of comments
* This program reads in system file auto.dta and gets summary statistics
clear // Remove data from memory
* The next code shows how to allow a single command to span two lines
sysuse ///
auto
summarize
```

For long commands, you can alternatively use the `#delimit` command. This changes the delimiter from the Stata default, which is a carriage return (that is, end of line), to a semicolon. This also permits more than one command on a single line. The following code changes the delimiter from the default to a semicolon and back to the default:

```
* Change delimiter from cr to semicolon and back to cr
#delimit ;
* More than one command per line and command spans more than one line;
clear; sysuse
auto; summarize;
#delimit cr
```

We recommend using `///` instead of changing the delimiter because the comment method produces more readable code.

#### 1.4.6 Different implementations of Stata

The different platforms for Stata share the same command syntax; however, commands can change across versions of Stata. For this book, we use Stata 17. To ensure that later versions of Stata will continue to work with our code, we include the `version 17` command near the beginning of the do-file.

Different implementations of Stata have different limits on, for example, the maximum number of variables in a dataset. These maximum possible values vary with the edition of Stata: Stata/BE, Stata/SE, or Stata/MP. The `help limits` command provides details on the limits for the

current implementation of Stata. The `query` and `creturn list` commands detail the current settings that may be below these limits.

Current settings can be increased or decreased with the `set` command, provided this does not lead to the maximum limit being exceeded. For example,

```
. set maxvar 10000
```

sets the maximum number of variables in a dataset to 10,000.

## 1.5 Scalars and matrices

Scalars can store a single number or a single string, and matrices can store several numbers or strings as an array. We provide a very brief introduction here, sufficient for use of the scalars and matrices in section [1.6](#).

### 1.5.1 Scalars

A scalar can store a single number or string. You can display the contents of a scalar by using the `display` command.

For example, to store the number  $2 \times 3$  as the scalar `a` and then display the scalar, we type

```
. * Scalars: Example  
. scalar a = 2*3  
. scalar b = "2 times 3 = "  
. display b a  
2 times 3 = 6
```

One common use of scalars, detailed in section [1.6](#), is to store the scalar results of estimation commands that can then be accessed for use in subsequent analysis. In section [1.7](#), we discuss the relative merits of using a scalar or a macro to store a scalar quantity. Scalars can have the same name as variables, in which case variables take precedence.

### 1.5.2 Matrices

Stata provides two distinct ways to use matrices, both of which store several numbers or strings as an array. One way is through Stata commands that have the `matrix` prefix. The second way is a much more powerful matrix programming language, Mata. These two methods are presented in, respectively, appendixes [A](#) and [B](#).

The following Stata code illustrates the definition of a specific  $2 \times 3$  matrix, the listing of the matrix, and the extraction and display of a specific element of the matrix.

```
. * Matrix commands: Example
. matrix define A = (1,2,3 \ 4,5,6)
. matrix list A
A[2,3]
    c1   c2   c3
r1    1    2    3
r2    4    5    6
. scalar c = A[2,3]
. display c
6
```

## 1.6 Using results from Stata commands

One goal of this book is to enable analysis that uses more than just official Stata commands and printed output. Much of this additional analysis entails further computations after using Stata commands.

### 1.6.1 Using results from the r-class command summarize

The Stata commands that analyze the data but do not estimate parameters are r-class commands. All r-class commands save their results in `r()`. The contents of `r()` vary with the command and are listed by typing `return list`.

As an example, we list the results stored after using `summarize`:

```
. * Illustrate use of return list for r-class command summarize
. summarize mpg
    Variable |       Obs        Mean   Std. dev.      Min      Max
              |      74     21.2973    5.785503     12      41
. return list
scalars:
          r(N) =  74
          r(sum_w) = 74
          r(mean) = 21.2972972972973
          r(Var) = 33.47204738985561
          r(sd) = 5.785503209735141
          r(min) = 12
          r(max) = 41
          r(sum) = 1576
```

There are eight separate results stored as Stata scalars with the names `r(N)`, `r(sum_w)`, ..., `r(sum)`. These are fairly obvious aside from `r(sum_w)`, which gives the sum of the weights. Several additional results are returned if the `detail` option to `summarize` is used; see [R] **summarize**.

The following code calculates and displays the range of the data:

```

. * Illustrate use of r()
. qui summarize mpg
. scalar range = r(max) - r(min)
. display "Sample range = " range
Sample range = 29

```

The results in `r()` disappear when a subsequent r-class or e-class command is executed. We can always save the value as a scalar. It can be particularly useful to save the sample mean.

```

. * Save a result in r() as a scalar
. scalar mpgmean = r(mean)

```

## 1.6.2 Using results from the e-class command regress

Estimation commands are e-class commands (or estimation-class commands), such as `regress`. The results are stored in `e()`, the contents of which you can view by typing `ereturn list`.

A leading example is `regress` for OLS regression. For example, after you type

regress mpg price weight						
Source	SS	df	MS	Number of obs	=	74
Model	1595.93249	2	797.966246	F(2, 71)	=	66.85
Residual	847.526967	71	11.9369995	Prob > F	=	0.0000
Total	2443.45946	73	33.4720474	R-squared	=	0.6531
				Adj R-squared	=	0.6434
				Root MSE	=	3.455
mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
price	-.0000935	.0001627	-0.57	0.567	-.000418	.0002309
weight	-.0058175	.0006175	-9.42	0.000	-.0070489	-.0045862
_cons	39.43966	1.621563	24.32	0.000	36.20635	42.67296

`ereturn list` yields

```

. * ereturn list after e-class command regress
. ereturn list

scalars:
    e(N) = 74
    e(df_m) = 2
    e(df_r) = 71
        e(F) = 66.84814256414501
        e(r2) = .6531446579233134
        e(rmse) = 3.454996314099513
        e(mss) = 1595.932492798133
        e(rss) = 847.5269666613265
        e(r2_a) = .6433740849070687
        e(l1) = -195.2169813478502
        e(l1_0) = -234.3943376482347
        e(rank) = 3

macros:
    e(cmdline) : "regress mpg price weight"
        e(title) : "Linear regression"
    e(marginsok) : "XB default"
        e(vce) : "ols"
        e(depvar) : "mpg"
        e(cmd) : "regress"
    e(properties) : "b V"
        e(predict) : "regres_p"
        e(model) : "ols"
    e(estat_cmd) : "regress_estat"

matrices:
    e(b) : 1 x 3
    e(V) : 3 x 3

functions:
    e(sample)

```

The key numeric output on sums of squares and degrees of freedom given in the analysis-of-variance table are stored as scalars. As an example of using scalar results, consider the calculation of  $R^2$ . The model sum of squares is stored in `e(mss)`, and the residual sum of squares is stored in `e(rss)`, so that

```

. * Use of e() where scalar
. scalar r2 = e(mss)/(e(mss)+e(rss))
. display "r-squared = " r2
r-squared = .65314466

```

The result is the same as the 0.6531 given in the original regression output.

The remaining numeric output are stored as matrices. Here we present methods to extract scalars from these matrices and manipulate them. Specifically, we obtain the OLS coefficient of `price` from the  $1 \times 3$  matrix `e(b)` and the estimated variance of this estimate from the  $3 \times 3$  matrix `e(V)`, and then we form the  $t$  statistic for testing whether the coefficient of `price` is 0:

```
. * Use of e() where matrix
. matrix best = e(b)
. scalar bprice = best[1,1]
. matrix Vest = e(V)
. scalar Vprice = Vest[1,1]
. scalar tprice = bprice/sqrt(Vprice)
. display "t statistic for H0: b_price = 0 is " tprice
t statistic for H0: b_price = 0 is -.57468079
```

The result is the same as the  $-0.57$  given in the original regression output.

Stata 16 and subsequent versions permit use of matrix subscripts and application of matrix functions returning scalars to `r()` and `e()` matrices. For example,

```
. * Direct use of matrix subscripts for e() matrices
. display e(b)[1,1]/sqrt(e(V)[1,1])
-.57468079
```

The results in `e()` disappear when a subsequent `e`-class command is executed. However, you can save the results by using `estimates store`, detailed in section [3.5.6](#).

## 1.7 Global and local macros

A macro is a string of characters that stands for another string of characters. For example, you can use the macro `xlist` in place of "price weight". This substitution can lead to code that is shorter, is easier to read, and can be easily adapted to similar problems.

Macros can be global or local. A global macro is accessible across Stata do-files or throughout a Stata session. A local macro can be accessed only within a given do-file or in the interactive session.

### 1.7.1 Global macros

Global macros are the simplest macro and are adequate for many purposes. We use global macros extensively throughout this book.

Global macros are defined with the `global` command. To access what was stored in a global macro, put the character `$` immediately before the macro name. For example, consider a regression of the dependent variable `mpg` on several regressors, where the global macro `xlist` is used to store the regressor list.

```
. * Global macro definition and use
. global xlist price weight
. regress mpg $xlist, noheader // $ prefix for global macro is necessary
```

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]
price	-.0000935	.0001627	-0.57	0.567	-.000418 .0002309
weight	-.0058175	.0006175	-9.42	0.000	-.0070489 -.0045862
_cons	39.43966	1.621563	24.32	0.000	36.20635 42.67296

Global macros are frequently used when fitting several different models with the same regressor list because they ensure that the regressor list is the same in all instances and they make it easy to change the regressor list. A single change to the global macro changes the regressor list in all instances.

A second example might be where several different models are fit, but we want to hold a key parameter constant throughout. For example, suppose

we obtain standard errors by using the bootstrap. Then we might define the global macro `nbreps` for the number of bootstrap replications. Exploratory data analysis might set `nbreps` to a small value such as 50 to save computational time, whereas final results set `nbreps` to an appropriately higher value such as 400.

A third example is to highlight key program parameters, such as the variable used to define the cluster if cluster-robust standard errors are obtained. By gathering all such global macros at the start of the program, we can know what the settings are for key program parameters.

### 1.7.2 Local macros

Local macros are defined with the `local` command. To access what was stored in the local macro, enclose the macro name in single quotes. These quotes differ from how they appear on this printed page. On most keyboards, the left quote is located at the upper left, under the tilde, and the right quote is located at the middle right, under the double quote.

As an example of a local macro, consider a regression of the `mpg` variable on several regressors. We define the local macro `xlist` and subsequently access its contents by enclosing the name in single quotes as '`xlist`'.

```
. * Local macro definition and use
. local xlist "price weight"
. regress mpg `xlist', noheader // Single quotes are necessary
```

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]
price	-.0000935	.0001627	-0.57	0.567	-.000418 .0002309
weight	-.0058175	.0006175	-9.42	0.000	-.0070489 -.0045862
_cons	39.43966	1.621563	24.32	0.000	36.20635 42.67296

The double quotes used in defining the local macro as a string are unnecessary, which is why we did not use them in the earlier global macro example. Using the double quotes does emphasize that a text substitution has been made. The single quotes in subsequent references to `xlist` are necessary.

We could also use a macro to define the dependent variable. For example,

```
. * Local macro definition without double quotes  
. local y mpg  
. regress `y` `xlist`, noheader
```

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]
price	-.0000935	.0001627	-0.57	0.567	-.000418 .0002309
weight	-.0058175	.0006175	-9.42	0.000	-.0070489 -.0045862
_cons	39.43966	1.621563	24.32	0.000	36.20635 42.67296

Note that here 'y' is not a variable with  $N$  observations. Instead, it is the string `mpg`. The `regress` command simply replaces 'y' with the text `mpg`, which in turn denotes a variable that has  $N$  observations.

We can also define a local macro through evaluation of a function. For example,

```
. * Local macro definition through function evaluation  
. local z = 2+2  
. display `z'  
4
```

leads to 'z' being the string 4. Using the equality sign when defining a macro causes the macro to be evaluated as an expression. For numerical expressions, using the equality sign stores the result of the expression and not the characters in the expression itself in the macro. For string assignments, it is best not to use the equality sign. This is especially true when storing lists of variables in macros.

Local macros are especially useful for programming in Stata; see appendix [A](#). Then, for example, you can use 'y' and 'x' as generic notation for the dependent variable and regressors, making the code easier to read.

Local macros apply only to the current program and have the advantage of no potential conflict with other programs. They are preferred to global macros, unless there is a compelling reason to use global macros.

### 1.7.3 Scalar or macro?

A macro can be used in place of a scalar, but a scalar is simpler. Furthermore, [P] **scalar** points out that using a scalar will usually be faster than using a macro because a macro requires conversion into and out of internal binary representation. This reference also gives an example where macros lead to a loss of accuracy because of these conversions.

One drawback of a scalar, however, is that the scalar is dropped whenever `clear all` is used. By contrast, a macro is still retained. Consider the following example:

```
. * Scalars disappear after clear all but macro does not
. global b 3
. local c 4
. scalar d = 5
. clear
. display $b _skip(3) `c`    // Display macros
3   4
. display d                  // Display the scalar
5
. clear all
. display $b _skip(3) `c`    // Display macros
3   4
. display d                  // Display the scalar
d not found
r(111);
```

Here the scalar `d` has been dropped after `clear all`, though not after `clear`.

We use global macros in this text because there are cases in which we want the contents of our macros to be accessible across do-files. A second reason for using global macros is that the required `$` prefix makes it clear that a global parameter is being used.

## 1.8 Looping commands

Loops provide a way to repeat the same command many times. We use loops in a variety of contexts throughout the book.

Stata has three looping constructs: `foreach`, `forvalues`, and `while`. The `foreach` construct loops over items in a list, where the list can be a list of variable names (possibly given in a macro) or a list of numbers. The `forvalues` construct loops over consecutive values of numbers. A `while` loop continues until a user-specified condition is not met.

We illustrate how to use these three looping constructs in creating the sum of four variables, where each variable is created from the uniform distribution. There are many variations in the way you can use these loop commands; see [P] **foreach**, [P] **forvalues**, and [P] **while**.

The `generate` command is used to create a new variable. The `runiform()` function provides a draw from the uniform distribution. Whenever random numbers are generated, we set the seed to a specific value with the `set seed` command so that subsequent runs of the same program lead to the same random numbers being drawn. We have, for example,

```
. * Make artificial dataset of 100 observations on 4 uniform variables
. clear
. set obs 100
Number of observations (_N) was 0, now 100.
. set seed 10101
. generate x1var = runiform()
. generate x2var = runiform()
. generate x3var = runiform()
. generate x4var = runiform()
```

We want to sum the four variables. The obvious way to do this is

. * Manually obtain the sum of four variables					
. generate sum = x1var + x2var + x3var + x4var					
. summarize sum					
Variable	Obs	Mean	Std. dev.	Min	Max
sum	100	1.935471	.5911068	.4193314	3.349523

We now present several ways to use loops to progressively sum these variables. Although only four variables are considered here, the same methods can potentially be applied to hundreds of variables.

### 1.8.1 The `foreach` loop

We begin by using `foreach` to loop over items in a list of variable names. Here the list is `x1var`, `x2var`, `x3var`, and `x4var`.

The variable ultimately created will be called `sum`. Because `sum` already exists, we need to first drop `sum` and then generate `sum=0`. The `replace sum=0` command collapses these two steps into one step, and the `quietly` prefix suppresses output stating that 100 observations have been replaced. Following this initial line, we use a `foreach` loop and additionally use `quietly` within the loop to suppress output following `replace`. The program is

. * foreach loop with a variable list					
. qui replace sum = 0					
. foreach var of varlist x1var x2var x3var x4var {					
2.       qui replace sum = sum + `var'					
3. }					
. summarize sum					
Variable	Obs	Mean	Std. dev.	Min	Max
sum	100	1.935471	.5911068	.4193314	3.349523

The result is the same as that obtained manually.

The preceding code is an example of a program (see appendix A) with the `{` brace appearing at the end of the first line and the `}` brace appearing on its own at the last line of the program. The numbers `2.` and `3.` do not

actually appear in the program but are produced as output. In the `foreach` loop, we refer to each variable in the variable list `varlist` by the local macro named `var`, so that ‘`var`’ with single quotes is needed in subsequent uses of `var`. The choice of `var` as the local macro name is arbitrary, and other names can be used. For a variable list, the word `varlist` is necessary. Types of lists other than variable lists are possible, in which case we use `numlist`, `newlist`, `global`, or `local`; see [P] **foreach**.

An attraction of using a variable list is that the method can be applied when variable names are not sequential. For example, the variable names could have been `incomehusband`, `incomewife`, `incomechild1`, and `incomechild2`.

### 1.8.2 The `forvalues` loop

A `forvalues` loop iterates over consecutive values. In the following code, we let the index be the local macro `i`, and ‘`i`’ with single quotes is needed in subsequent uses of `i`. The program

```
. * forvalues loop to create a sum of variables
. qui replace sum = 0
. forvalues i = 1/4 {
    2.      qui replace sum = sum + x`i`var
    3. }
. summarize sum
```

Variable	Obs	Mean	Std. dev.	Min	Max
sum	100	1.935471	.5911068	.4193314	3.349523

produces the same result.

The choice of the name `i` for the local macro was arbitrary. In this example, the increment is one, but you can use other increments. For example, if we use `forvalues i = 1(2)11`, then the index goes from 1 to 11 in increments of 2.

### 1.8.3 The `while` loop

A `while` loop continues until a condition is no longer met. This method is used when `foreach` and `forvalues` cannot be used. For completeness, we apply it to the summing example.

In the following code, the local macro `i` is initialized to 1 and then incremented by 1 in each loop; looping continues, provided that  $i \leq 4$ .

```
. * While loop and local macros to create a sum of variables
. qui replace sum = 0
. local i 1
. while `i' <= 4 {
2.     qui replace sum = sum + x`i'var
3.     local i = `i' + 1
4. }
. summarize sum
```

Variable	Obs	Mean	Std. dev.	Min	Max
sum	100	1.935471	.5911068	.4193314	3.349523

#### 1.8.4 The `continue` command

The `continue` command provides a way to prematurely cease execution of the current loop iteration. This may be useful if, for example, the loop includes taking the log of a number and we want to skip this iteration if the number is negative. Execution then resumes at the start of the next loop iteration, unless the `break` option is used. For details, see `help continue`.

## 1.9 Mata and Python in Stata

Stata can enact commands written in programming languages other than Stata. In particular, Stata has a separate matrix programming language, Mata. One can enter and exit Mata from Stata. Additionally, commands in Mata can be implemented within Stata using the prefix `mata:`, and commands in Stata can be implemented in Mata using the prefix `stata:`. Appendix [B](#) details the Mata programming language, and many Mata examples are given in this book.

Stata version 16 introduced the `python` command, which provides similar facility for the Python language. As of Stata 17, Stata can be called from Python via the `pystata` Python package.

More generally, the `plugin` command permits execution within Stata of compiled libraries written in other programming languages. The `javacall` command handles the special case of Java plugins. With the `java` command, Java code can be executed directly from within Stata.

## **1.10 Some useful commands**

We have mentioned only a few Stata commands. See [U] **28 Commands everyone should know** for a list of key commands that everyone will find useful.

## 1.11 Template do-file

The following do-file provides a template. It captures most of the features of Stata presented in this chapter, aside from looping commands.

```
* 1. Program name
* mus201p2template.do written 12/01/2021 is a template do-file
* 2. Write output to a log file
log using mus 201p2template.txt, text replace
* 3. Stata version
version 17          // So will still run in a later version of Stata
* 4. Program explanation
* This illustrative program creates 100 uniform variates
* 5. Change Stata default settings - one example is given
set linesize 82      // Set the maximum width of Stata output
* 6. Set program parameters using global macros
global numobs 100
local seed 10101
local xlist xvar
* 7. Generate data and summarize
set obs $numobs
set seed `seed'
generate xvar = runiform()
generate yvar = xvar^2
summarize
* 8. Demonstrate use of results stored in r()
summarize xvar
display "Sample range = " r(max)-r(min)
regress yvar `xlist', vce(robust)
scalar r2 = e(mss)/(e(mss)+e(rss))
display "r-squared = " r2
* 9. Close output file and exit Stata
log close
exit, clear
```

## 1.12 Community-contributed commands

We make extensive use of community-contributed commands. These are freely available ado-files (see section [A.2.8](#)) that are easy to install, provided you are connected to the Internet and, for computer lab users, that the computer lab places no restriction on adding components to Stata. They are then executed in the same way as Stata commands.

As an example, consider instrumental-variables (IV) estimation. In some cases, we know which community-contributed commands we want. For example, a leading community-contributed command for IV is `ivreg2`, and we type `search ivreg2` to get it. More generally, we can type the broader command

```
. search instrumental variables  
(output omitted)
```

This gives information on IV commands available both within Stata and packages available on the web, provided you are connected to the Internet.

Many entries are provided, often with several potential community-contributed commands and several versions of a given community-contributed command. The best place to begin can be an SJ article because this code is more likely to have been closely vetted for accuracy and written in a way suited to a range of applications. The listing from the `search` command includes

```
SJ-7-4 st0030_3 . . . Enhanced routines for IV/GMM estimation and testing  
. . . . . C. F. Baum, M. E. Schaffer, and S. Stillman  
(help ivactest, ivendog, ivhettest, ivreg2, ivreset,  
overid, ranktest if installed)  
Q4/07 SJ 7(4):465--506  
extension of IV and GMM estimation addressing hetero-  
skedasticity- and autocorrelation-consistent standard  
errors, weak instruments, LIML and k-class estimation,  
tests for endogeneity and Ramsey's regression  
specification-error test, and autocorrelation tests  
for IV estimates and panel-data IV estimates
```

The entry means that it is the third revision of the package (`st0030_3`), and the package is discussed in detail in SJ, volume 7, number 4 (SJ-7-4).

By left-clicking on the highlighted text `st0030_3` on the first line of the entry, you will see a new window with title, description and authors, and installation files for the package. By left-clicking on the help files, you can obtain information on the commands. By left-clicking on (`click here to install`), you will install the files into an ado-directory.

Many community-contributed programs are stored at the Statistical Software Components website. These can be directly installed using the `ssc install` command. The more general `net` command can be used to download Stata packages from any source—Internet or physical media.

Community-contributed commands are periodically updated. Entering `adoupdate` determines the current update level and provides the option to install updates to community-contributed files.

## 1.13 Additional resources

For first-time users, [GS] *Getting Started with Stata* is very helpful, along with analyzing an example dataset such as `auto.dta` interactively in Stata. The next useful manual is [U] *Users Guide*, especially the early chapters. For further resources, see section [1.2.2](#).

## 1.14 Exercises

1. Find information on the estimation method `clogit` using `help` and `search`. Comment on the relative usefulness of these search commands.
2. Download the Stata example dataset `auto.dta`. Obtain summary statistics for `mpg` and `weight` according to whether the car type is foreign (use the `by foreign:` prefix). Comment on any differences between foreign and domestic cars. Then, `regress mpg` on `weight` and `foreign`. Comment on any difference for foreign cars.
3. Write a do-file to repeat the previous question. This do-file should include a log file. Run the do-file, and then use a text editor to view the log file.
4. Using `auto.dta`, obtain summary statistics for the `price` variable. Then, use the results stored in `r()` to compute a scalar, `cv`, equal to the coefficient of variation (the standard deviation divided by the mean) of `price`.
5. Using `auto.dta`, `regress mpg` on `price` and `weight`. Then, use the results stored in `e()` to compute a scalar, `r2adj`, equal to  $\bar{R}^2$ . The adjusted  $R^2$  equals  $R^2 - (1 - R^2)(K - 1)/(N - K)$ , where  $N$  is the number of observations and  $K$  is the number of regressors including the intercept. Also, use the results stored in `e()` to calculate a scalar, `tweight`, equal to the  $t$  statistic to test that the coefficient of `weight` is zero.
6. Using `auto.dta`, define a global macro named `varlist` for a variable list with `mpg`, `price`, and `weight`, and then obtain summary statistics for `varlist`. Repeat this exercise for a local macro named `varlist`.
7. Using `auto.dta`, use a `foreach` loop to create a variable, `total`, equal to the sum of `headroom` and `length`. Confirm by using `summarize` that `total` has a mean equal to the sum of the means of `headroom` and `length`.
8. Create a simulated dataset with 100 observations on two random variables that are each drawn from the uniform distribution. Use a seed of 12345. In theory, these random variables have a mean of 0.5 and a variance of 1/12. Does this appear to be the case here?



# **Chapter 2**

## **Data management and graphics**

## 2.1 Introduction

The starting point of an empirical investigation based on microeconomic data is the collection and preparation of a relevant dataset. The primary sources are often government surveys and administrative data. We assume the researcher has such a primary dataset and do not address issues of survey design and data collection. Even given primary data, it is rare that they will be in a form that is exactly what is required for ultimate analysis.

The process of transforming original data to a form that is suitable for econometric analysis is referred to as data management. This is typically a time-intensive task that has important implications for the quality and reliability of modeling carried out at the next stage.

This process usually begins with a data file or files containing basic information extracted from a census or a survey. They are often organized by data record for a sampled entity such as an individual, a household, or a firm. Each record or observation is a vector of data on the qualitative and quantitative attributes of each individual. Typically, the data need to be cleaned up and recoded, and data from multiple sources may need to be combined. The focus of the investigation might be a particular group or subpopulation, for example, employed women, so that a series of criteria need to be used to determine whether a particular observation in the dataset is to be included in the analysis sample.

In this chapter, we present the tasks involved in data preparation and management. These include reading in and modifying data, transforming data, merging data, checking data, and selecting an analysis sample. The rest of the book focuses on analyzing a given sample, though special features of handling panel data and multinomial data are given in the relevant chapters.

## 2.2 Types of data

All data are ultimately stored in a computer as a sequence of 0s and 1s because computers operate on binary digits, or bits, that are either 0 or 1. There are several different ways to do this, with potential to cause confusion.

### 2.2.1 Text or ASCII data

A standard text format is ASCII, an acronym for American Standard Code for Information Interchange. Regular ASCII represents  $2^7 = 128$ , and extended ASCII represents  $2^8 = 256$  different digits, letters (uppercase and lowercase), and common symbols and punctuation marks. In either case, eight bits (called a byte) are used. As examples, 1 is stored as 00110001, 2 is stored as 00110010, 3 is stored as 00110011, A is stored as 01010001, and a is stored as 00110001. A text file that is readable on a computer screen is stored in ASCII.

A leading text-file example is a spreadsheet file that has been stored as a “comma-separated values” file, usually a file with the `.csv` extension. Here a comma is used to separate each data value; however, more generally, other separators can be used.

Text-file data can also be stored as fixed-width data. Then no separator is needed provided we use the knowledge that, say, columns 1–7 have the first data entry, columns 8–9 have the second data entry, and so on.

Text data can be numeric or nonnumeric. The letter *a* is clearly nonnumeric, but depending on the context, the number 3 might be numeric or nonnumeric. For example, the number 3 might represent the number of doctor visits (numeric) or be part of a street address, such as 3 Main Street (nonnumeric).

### 2.2.2 Internal numeric data

When data are numeric, the computer stores them internally using a format different from text to enable application of arithmetic operations and to reduce storage. The two main types of numeric data are integer and floating point.

Because computers work with 0s and 1s (a binary digit or bit), data are stored in base-2 approximations to their base-10 counterparts.

For integer data, the exact integer can be stored. The size of the integer stored depends on the number of bytes used, where a byte is eight bits. For example, if one byte is used, then in theory  $2^8 = 256$  different integers could be stored, such as  $-127, -126, \dots, 127, 128$ .

Noninteger data, or often even integer data, are stored as floating-point data. Standard floating-point data are stored in four bytes, where the first bit may represent the sign, the next 8 bits may represent the exponent, and the remaining 23 bits may represent the digits. Although all integers have an exact base-2 representation, not all base-10 numbers do. For example, the base-10 number 0.1 is 0.00011 in base 2. Thus, the more bytes in the base-2 approximation, the more precisely it approximates the base-10 number. Double-precision floating-point data use 8 bytes, have about 16 digits precision (in base 10), and are sufficiently accurate for most statistical calculations.

Care is needed, however, in commands that rely on numerical equality. Because data are usually in base 10, but calculations are in base 2, numerical computation need not be exact. For example, while it is well known that  $(0.3 + 0.6 + 0.1) = (0.3 + 0.1 + 0.6)$ , in fact the different orderings of the components of the sum lead to different results. We have

```
. * Example of numerical error
. display %25.20f 0.3+0.6+0.1
>      _n %25.20f 0.3+0.1+0.6
>      _n %25.20f (0.3+0.6+0.1)-(0.3+0.1+0.6)
    0.9999999999999988898
    1.0000000000000000000000000
   -0.00000000000000011102
```

For more details, see [U] **13.12 Precision and problems therein**.

Stata has the numeric storage types listed in table [2.1](#): three are integer and two are floating point.

**Table 2.1.** Stata's numeric storage types

Storage type	Bytes	Minimum	Maximum
<code>byte</code>	1	−127	100
<code>int</code>	2	−32,767	32,740
<code>long</code>	4	−2,147,483,647	2,147,483,620
<code>float</code>	4	$-1.70141173319 \times 10^{38}$	$1.70141173319 \times 10^{38}$
<code>double</code>	8	$-8.9984656743 \times 10^{307}$	$8.9984656743 \times 10^{307}$

These internal data types have the advantage of taking fewer bytes to store the same amount of data. For example, the integer 123456789 takes up 9 bytes if stored as text but only 4 bytes if stored as an integer (`long`) or floating point (`float`). For large or long numbers, the savings can clearly be much greater. The Stata default is for floating-point data to be stored as `float` and for computations to be stored as `double`.

The `compress` command reduces the size of datasets by automatically converting data where appropriate to a storage type that uses fewer bytes. For example, a variable stored as `float` may be converted to `int` or `byte`. This command is particularly useful for very large datasets.

Data read into Stata are stored using these various formats, and Stata data files (`.dta`) use these formats. One disadvantage is that numbers in internal-storage form cannot be read in the same way that text can; we need to first reconvert them to a text format. A second disadvantage is that it is not always easy to transfer data in internal format across packages, though the Stata `import` and `export` commands make this easy for a number of formats including Excel and SAS.

It is much easier to transfer data that is stored as text data. Downsides, however, are an increase in the size of the dataset compared with the same dataset stored in internal numeric form, and possible loss of precision in converting floating-point data to text format.

### 2.2.3 String data

Nonnumeric data in Stata are recorded as strings, typically enclosed in double quotes, such as “3 Main Street”. The storage type `str20`, for example, states

that the data should be stored as a string of length 20 characters.

In this book, we focus on numeric data and seldom use strings. Stata has many commands for working with strings. Two useful commands are `destring`, which converts string data to integer data, and `tostring`, which does the reverse.

## 2.2.4 Formats for displaying numeric data

Stata output and text files written by Stata format data for readability. The format is automatically chosen by Stata but can be overridden.

The most commonly used format is the `f` format, or the fixed format. An example is `%7.2f`, which means the number will be right justified and fill 7 columns with 2 digits after the decimal point. For example, 123.321 is represented as 123.32.

The format type always begins with `%`. The default of right justification is replaced by left justification if an optional `-` follows. Then follows an integer for the width (number of columns), a period (`.`), an integer for the number of digits following the decimal point, and an `e` or an `f` or a `g` for the format used. An optional `c` at the end leads to comma format.

The usual format is the `f` format, or fixed format, for example, 123.32. The `e`, or exponential, format (scientific notation) is used for very large or small numbers, for example, 1.23321e+02. The `g`, or general format, leads to `e` or `f` being chosen by Stata in a way that will work well regardless of whether the data are very large or very small. In particular, the format `%#. (#-1)g` will vary the number of columns after the decimal point optimally. For example, `%8.7g` will present a space followed by the first six digits of the number and the appropriately placed decimal point.

To see the various possible formats, enter `help format`.

## 2.3 Inputting data

The starting point is the computer-readable file that contains the raw data. Where large datasets are involved, this is typically either a text file or the output of another computer program, such as Excel, SAS, or even Stata.

### 2.3.1 General principles

For a discussion of initial use of Stata, see chapter [1](#).

To replace any existing dataset in memory, you need to first clear the current dataset.

```
. * Remove current dataset from memory  
. clear
```

This removes data and any associated value labels from memory. If you are reading in data from a Stata dataset, you can instead use the `clear` option with the `use` command. Various arguments of `clear` lead to additional removal from memory of matrices, scalars, constraints, clusters, stored results, programs, frames, collections of results, and sersets and Mata functions, as well as to closing all open files and postfiles, clearing the class system, closing any open Graph windows and dialog boxes, and resetting all timers to zero. The `clear all` command removes all of these.

Various commands are used to read in data, depending on the format of the file being read. These commands, discussed in detail in the rest of this section, include the following:

- `use` to read a Stata dataset (with extension `.dta`)
- `edit` and `input` to enter data from the Data Editor or the keyboard
- variations of the `import` command, detailed below, to read data in nontext formats such as Microsoft Excel worksheets
- `odbc` to read data from Open Database Connectivity sources
- `jdbc` to read data using Java Database Connectivity
- `import delimited` to read comma-separated or tab-separated text data created by a spreadsheet

- `infile` to read unformatted (free format) separated text data
- `infix` to read fixed-column format text data
- `infile` to read fixed-column format text data (more flexible than `infix`)

As soon as data are input into Stata, you should save the data as a Stata dataset. For example,

```
. * Save data as a Stata dataset
. save mydata, replace
(output omitted)
```

The `replace` option will replace any existing dataset with the same name. If you do not want this to happen, then do not use the option.

To check that data are read in correctly, list the first few observations, use `describe`, and obtain the summary statistics.

```
. * Quick check that data are read in correctly
. list in 1/5      // List the first five observations
(output omitted)
. describe        // Describe the variables
(output omitted)
. summarize       // Descriptive statistics for the variables
(output omitted)
```

Examples illustrating the output from `describe` and `summarize` are given in sections [2.4.1](#) and [3.2](#).

### 2.3.2 Inputting data already in Stata format

Data in the Stata format are stored with the `.dta` extension, for example, `mydata.dta`. Then the data can be read in with the `use` command. For example,

```
. * Read in existing Stata dataset
. use c:\research\mydata, clear
```

The `clear` option removes any data currently in memory, even if the current data have not been saved, enabling the new file to be read into memory.

If Stata is initiated from the current directory, then we can more simply type

```
. * Read in dataset in current directory  
. use mydata, clear
```

The `use` command also works over the Internet, provided that your computer is connected. For example, you can obtain an extract from the 1980 U.S. Census by typing

```
. * Read in dataset from an Internet website  
. use http://www.stata-press.com/data/r17/census  
(1980 Census data by state)
```

### 2.3.3 Inputting data from the keyboard

The `input` command enables data to be typed in from the keyboard. It assumes that data are numeric. If instead data are characters, then `input` should additionally define the data as a string and give the string length. For example,

```
. * Data input from keyboard using input  
. clear  
. input str20 name age female income  
          name      age     female    income  
1.   "Barry"  25  0 40.990  
2.   "Carrie" 30  1 37.000  
3.   "Gary"   31  0 48.000  
4. end
```

The quotes here are not necessary; we could use `Barry` rather than "`Barry`". If the name includes a space, such as "`Barry Jr`", then double quotes are needed; otherwise, `Barry` would be read as a string, and then `Jr` would be read as a number, leading to a program error.

To check that the data are read in correctly, we use the `list` command. Here we add the `clean` option, which lists the data without divider and separator lines.

```
. list, clean
      name    age   female   income
1.  Barry     25        0     40.99
2. Carrie    30        1      37
3. Gary      31        0      48
```

### 2.3.4 Inputting nontext data

By nontext data, we mean data that are stored in the internal code of a software package other than Stata. It is easy to establish whether a file is a nontext file by viewing the file using a text editor. If strange characters appear, then the file is a nontext file. An example is an Excel .xls file.

Stata supports several special formats. Specifically,

- the `import dbase` command reads a version III or IV dBase (.dbf) file.
- the `import excel` command reads worksheets from Microsoft Excel (.xls and .xlsx) workbooks.
- the `import fred` command reads individual Federal Reserve Economics Data series.
- the `import haver` command reads individual Haver Analytics database series.
- the `import sas` command reads version 7 SAS (.sas7bdat) files.
- the `import sasxport5` and `import sasxport8` commands read SAS XPORT Transport format files.
- the `import spss` command reads version 7 SAS (.sav and .zsav) files.
- the `odbc` command reads Open Database Connectivity data files.
- the `jdbc` command reads data using Java Database Connectivity
- the `spshape2dta` command for geospatial data translates .dbf and .shp files of a shapefile into two Stata datasets.

Here we detail the `import excel` command. The default is to import the first worksheet in the Excel file. The following example imports a selected sheet, here called `grades`, in which the sheet name needs to be provided. The `firstrow` option is used if the first row of the worksheet has variable names.

```
. * Read in a specific worksheet from an Excel workbook
. import excel myworkbook.xlsx, sheet("grades") firstrow clear
```

The commercial software package Stat/Transfer supports file conversion, such as from MATLAB to Stata, for many file formats.

### 2.3.5 Inputting text data from a spreadsheet

The `import delimited` command, which supplants the earlier `insheet` command, reads data that are saved by a spreadsheet or database program as comma-separated or tab-separated text data. For example, `mus202file1.csv`, a file with comma-separated values, has the following data:

```
name,age,female,income
Barry,25,0,40.990
Carrie,30,1,37.000
Gary,31,0,48.000
```

To read these data, we use `import delimited`. Thus,

```
. * Read data from a .csv file that includes variable names using import delimited
. clear
. import delimited using mus202file1.csv
(encoding automatically selected: ISO-8859-2)
(4 vars, 3 obs)
. list, clean
      name    age    female    income
1.    Barry     25        0     40.99
2.    Carrie    30        1      37
3.    Gary      31        0      48
```

Stata automatically recognized the `name` variable to be a string variable, the `age` and `female` variables to be integer, and the `income` variable to be floating point.

A major advantage of `import delimited` is that it can read in a text file that includes variable names as well as data, making mistakes less likely. There are some limitations, however. The `import delimited` command is restricted to files with a single observation per line. And the data must be comma-separated or tab-separated, but not both. It cannot be space-separated, but other delimiters can be specified by using the `delimiter` option.

The first line with variable names is optional. Let `mus202file2.csv` be the same as the original file, except without the header line:

```
Barry,25,0,40.990  
Carrie,30,1,37.000  
Gary,31,0,48.000
```

The `import delimited` command still works. By default, the variables read in are given the names `v1`, `v2`, `v3`, and `v4`. Alternatively, you can assign more meaningful names in `import delimited`. For example,

```
. * Read data from a .csv file without variable names, and assign names  
. clear  
. import delimited name age female income using mus202file2.csv  
(encoding automatically selected: ISO-8859-2)  
(4 vars, 3 obs)
```

### 2.3.6 Inputting text data in free format

The `infile` command reads free-format text data that are space-separated, tab-separated, or comma-separated.

We again consider `mus202file2.csv`, which has no header line. Then

```
. * Read data from free-format text file using infile  
. clear  
. infile str20 name age female income using mus202file2.csv  
(3 observations read)  
. list, clean
```

	name	age	female	income
1.	Barry	25	0	40.99
2.	Carrie	30	1	37
3.	Gary	31	0	48

By default, `infile` reads in all data as numbers that are stored as floating point. This causes obvious problems if the original data are string. By inserting `str20` before `name`, the first variable is instead a string that is stored as a string of at most 20 characters.

For `infile`, a single observation is allowed to span more than one line, or there can be more than one observation per line. Essentially, every fourth

entry after `Barry` will be read as a string entry for `name`, every fourth entry after `25` will be read as a numeric entry for `age`, and so on.

The `infile` command is the most flexible command to read in data and will also read in fixed-format data.

### 2.3.7 Inputting text data in fixed format using `infix`

The `infix` command reads fixed-format text data that are in fixed-column format. For example, suppose `mus202file3.txt` contains the same data as before, except without the header line and with the following fixed format:

```
Barry      250 40.990
Carrie     301 37.000
Gary       310 48.000
```

Here columns 1–10 store the `name` variable, columns 11–12 store the `age` variable, column 13 stores the `female` variable, and columns 14–20 store the `income` variable.

Note that a special feature of fixed-format data is that there need be no separator between data entries. For example, for the first observation, the sequence `250` is not `age` of `250` but is instead two variables: `age = 25` and `female = 0`. It is easy to make errors when reading fixed-format data.

To use `infix`, we need to define the columns in which each entry appears. There are a number of ways to do this. For example,

```
. * Read data from fixed-format text file using infix
. clear
. infix str20 name 1-10 age 11-12 female 13 income 14-20 using mus202file3.txt
(3 observations read)
. list, clean
      name    age    female    income
1.   Barry     25        0     40.99
2.   Carrie    30        1      37
3.   Gary      31        0      48
```

As with `infile`, we include `str20` to indicate that `name` is a string rather than a number.

A single observation can appear on more than one line. Then we use the symbol / to skip a line or use the entry 2:, for example, to switch to line 2. For example, suppose `mus202file4.txt` is the same as `mus202file3.txt`, except that `income` appears on a separate second line for each observation in columns 1–7. Then,

```
. * Read data using infix where an observation spans more than one line  
. clear  
. infix str20 name 1-10 age 11-12 female 13 2: income 1-7 using mus202file4.txt  
(3 observations read)
```

### 2.3.8 Inputting text data in fixed format using infile and a dictionary

For simple fixed-format text datasets, the `infix` command is adequate. For more complicated fixed-format text datasets, the format for the data being read in can be stored in a dictionary file, a text file created by a word processor, or an editor. Details are provided in [D] **infile (fixed format)**. Suppose this file is called `mus202dict.dct` and the data are in file `mus202file3.txt`. Then we type

```
. * Read in data with dictionary file  
. infile using mus202dict.dct, using(mus202file3.txt)
```

where the dictionary file `mus202dict.dct` provides variable names and formats.

### 2.3.9 Common pitfalls

It can be surprisingly difficult to read in data. With fixed-format data, wrong column alignment leads to errors. Data can unexpectedly include string data, perhaps with embedded blanks. Missing values might be coded as not applicable, causing problems if a numeric value is expected. An observation can span several lines when a single line was erroneously assumed.

It is possible to read a dataset into Stata without Stata issuing an error message; no error message does not mean that the dataset has been successfully read in. For example, transferring data from one computer type to another, such as a file transfer using File Transfer Protocol, can lead to an additional carriage return, or *Enter*, being typed at the end of each line. Then

`infix` reads the dataset as containing one line of data, followed by a blank line, then another line of data, and so on. The blank lines generate extraneous observations with missing values.

You should always perform checks, such as using `list` and `summarize`. Always view the data before beginning analysis.

### 2.3.10 Outputting data

Stata datasets can be output to other formats. Text data files can be created using the `export delimited` command, for data separated using a comma or other separator, or the more flexible `outfile` command (see section [2.4.8](#)), which creates both separated data files and fixed-format data files.

Nontext special format data files can be created using commands `export dbase`, `export excel`, `odbc`, `jdbc`, `export sasxport5`, and `export sasxport8`.

## 2.4 Data management

Once the data are read in, there can be considerable work in cleaning up the data, transforming variables, and selecting the final sample. All data management tasks should be recorded, dated, and saved. The existence of such a record makes it easier to track changes in definitions and eases the task of replication. By far, the easiest way to do this is to have the data management manipulations stored in a do-file rather than to use commands interactively. We assume that a do-file is used.

### 2.4.1 Panel Study of Income Dynamics example

Data management is best illustrated using a real-data example. Typically, one needs to download the entire original dataset and an accompanying document describing the dataset. For some major commonly used datasets, however, there may be cleaned-up versions of the dataset, simple data-extraction tools, or both.

Here we obtain a very small extract from the 1992 Individual-Level data from the Panel Study of Income Dynamics (PSID), a U.S. longitudinal survey conducted by the University of Michigan. The extract was downloaded from the Data Center at the website <https://psidonline.isr.umich.edu/>, using interactive tools to select just a few variables. The extracted sample was restricted to men aged 30–50 years. The output conveniently included a Stata do-file in addition to the text data file. Additionally, a codebook describing the variables selected was provided. The data download included several additional variables that enable unique identifiers and provide sample weights. These should also be included in the final dataset but, for brevity, have been omitted below.

Reading the text dataset `mus202psid92m.txt` using a text editor reveals that the first two observations are

```
4^ 3^ 1^ 2^ 1^ 2482^ 1^ 10^ 40^ 9^ 22000^ 2340  
4^ 170^ 1^ 2^ 1^ 6974^ 1^ 10^ 37^ 12^ 31468^ 2008
```

The data are text data delimited by the symbol ^.

Several methods could be used to read the data, but the simplest is to use `import delimited`. This is especially simple given the provided do-file. The `mus202psid92m.do` file contains the following information:

```
. * Commands to read in data from PSID extract in a delimited file
. type mus202psid92m.do
* mus202psid92m.do
clear
#delimit ;
* PSID DATA CENTER ****
JOBID : 10654
DATA_DOMAIN : PSID
USER_WHERE : ER32000=1 and ER30736 ge 30 and ER
FILE_TYPE : All Individuals Data
OUTPUT_DATA_TYPE : ASCII Data File
STATEMENTS : STATA Statements
CODEBOOK_TYPE : PDF
N_OF_VARIABLES : 12
N_OF_OBSERVATIONS: 4290
MAX_REC_LENGTH : 56
DATE & TIME : November 3, 2003 @ 0:28:35
*****
;
import delimited
er30001 er30002 er32000 er32022 er32049 er30733 er30734 er30735 er30736
er30748 er30750 er30754
using mus202psid92m.txt, delim("") clear
;
destring, replace ;
label variable er30001 "1968 INTERVIEW NUMBER" ;
label variable er30002 "PERSON NUMBER" 68" ;
label variable er32000 "SEX OF INDIVIDUAL" ;
label variable er32022 "# LIVE BIRTHS TO THIS INDIVIDUAL" ;
label variable er32049 "LAST KNOWN MARITAL STATUS" ;
label variable er30733 "1992 INTERVIEW NUMBER" ;
label variable er30734 "SEQUENCE NUMBER" 92" ;
label variable er30735 "RELATION TO HEAD" 92" ;
label variable er30736 "AGE OF INDIVIDUAL" 92" ;
label variable er30748 "COMPLETED EDUCATION" 92" ;
label variable er30750 "TOT LABOR INCOME" 92" ;
label variable er30754 "ANN WORK HRS" 92" ;
#delimit cr; // Change delimiter to default cr
```

To read the data, we need only `import delimited`. The code separates commands using the delimiter ; rather than the default `cr` (the *Enter* key or carriage return) to enable comments and commands that span several lines.

The `destring` command, unnecessary here, converts any string data into numeric data. For example, \$1,234 would become 1234. The `label` variable command provides a longer description of the data that will be reproduced by using `describe`.

Executing this code yields output that includes the following:

```
(12 vars, 4290 obs)
. destring, replace ;
er30001 already numeric; no replace
  (output omitted)
er30754 already numeric; no replace
```

The statement `already numeric` is output for all variables because all the data in `mus202psid92m.txt` are numeric.

The `describe` command provides a description of the data:

```
. * Data description
. describe
Contains data
Observations:          4,290
Variables:             12
```

Variable name	Storage type	Display format	Value label	Variable label
er30001	int	%8.0g		1968 INTERVIEW NUMBER
er30002	int	%8.0g		PERSON NUMBER 68
er32000	byte	%8.0g		SEX OF INDIVIDUAL
er32022	byte	%8.0g		# LIVE BIRTHS TO THIS INDIVIDUAL
er32049	byte	%8.0g		LAST KNOWN MARITAL STATUS
er30733	int	%8.0g		1992 INTERVIEW NUMBER
er30734	byte	%8.0g		SEQUENCE NUMBER 92
er30735	byte	%8.0g		RELATION TO HEAD 92
er30736	byte	%8.0g		AGE OF INDIVIDUAL 92
er30748	byte	%8.0g		COMPLETED EDUCATION 92
er30750	long	%12.0g		TOT LABOR INCOME 92
er30754	int	%8.0g		ANN WORK HRS 92

Sorted by:

Note: Dataset has changed since last saved.

The `summarize` command provides descriptive statistics:

```
. * Data summary
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
er30001	4,290	4559.2	2850.509	4	9308
er30002	4,290	60.66247	79.93979	1	227
er32000	4,290	1	0	1	1
er32022	4,290	21.35385	38.20765	1	99
er32049	4,290	1.699534	1.391921	1	9
er30733	4,290	4911.015	2804.8	1	9829
er30734	4,290	3.179487	11.4933	1	81
er30735	4,290	13.33147	12.44482	10	98
er30736	4,290	38.37995	5.650311	30	50
er30748	4,290	14.87249	15.07546	0	99
er30750	4,290	27832.68	31927.35	0	999999
er30754	4,290	1929.477	899.5496	0	5840

Satisfied that the original data have been read in carefully, we proceed with cleaning the data.

## 2.4.2 Naming and labeling variables

Just as the Data Editor can be used to input and manage data, the Variables Manager can be used to manage the properties of variables, such as their names and labels. We use Stata commands below to rename and label variables, but we could also have used the Variables Manager.

The first step is to give more meaningful names to variables by using the `rename` command. We do so just for the variables used in subsequent analysis.

```
. * Rename variables
. rename er32000 sex
. rename er30736 age
. rename er30748 education
. rename er30750 earnings
. rename er30754 hours
```

The renamed variables retain the descriptions that they were originally given. Some of these descriptions are unnecessarily long, so we use `label`

variable to shorten output from commands, such as `describe`, that give the variable labels.

```
. * Relabel some of the variables
. label variable age "Age of individual"
. label variable education "Completed education"
. label variable earnings "Total labor income"
. label variable hours "Annual work hours"
```

For categorical variables, it can be useful to explain the meanings of the variables. For example, from the codebook discussed in section [2.4.4](#), the `er32000` variable takes on the value 1 if male and 2 if female. We may prefer that the output of variable values uses a label in place of the number. These labels are provided by using `label define` together with `label values`.

```
. * Define the label gender for the values taken by variable sex
. label define gender 1 male 2 female
. label values sex gender
. list sex in 1/2, clean

      sex
1.   male
2.   male
```

After renaming, we obtain

```
. * Data summary of key variables after renaming
. summarize sex age education earnings hours
```

Variable	Obs	Mean	Std. dev.	Min	Max
sex	4,290	1	0	1	1
age	4,290	38.37995	5.650311	30	50
education	4,290	14.87249	15.07546	0	99
earnings	4,290	27832.68	31927.35	0	999999
hours	4,290	1929.477	899.5496	0	5840

Data exist for these variables for all 4,290 sample observations. The data have  $30 \leq \text{age} \leq 50$  and `sex = 1` (male) for all observations, as expected. The maximum value for `earnings` is \$999,999, an unusual value that most likely indicates top-coding. The maximum value of `hours` is quite high and may also indicate top-coding ( $365 \times 16 = 5840$ ). The maximum value of 99 for `education` is clearly erroneous; the most likely explanation is that this is

a missing-value code because numbers such as 99 or –99 are often used to denote a missing value.

### 2.4.3 Viewing data

The standard commands for viewing data are `summarize`, `list`, and `tabulate`.

We have already illustrated the `summarize` command. Additional statistics, including key percentiles and the five largest and smallest observations, can be obtained by using the `detail` option; see section [3.2.4](#).

The `list` command can list every observation, too many in practice. But you could list just a few observations:

```
. * List first 2 observations of two of the variables
. list age hours in 1/2, clean
      age    hours
1.    40    2340
2.    37    2008
```

The `list` command with no variable list provided will list all the variables. The `clean` option eliminates dividers and separators.

The `format` command sets the display format of one or more variables. For example,

```
. * Change display format of variable hours
. format %15.2f hours
. list age hours in 1/2, clean
      age    hours
1.    40    2340.00
2.    37    2008.00
```

The `count` command counts the number of observations satisfying a given criterion. For example, to count the number of persons with less than 12 years of education, we type

```
. * Count number of observations satisfying a given condition
. count if education < 12
934
```

The `tabulate` command lists each distinct value of the data and the number of times it occurs. It is useful for data that do not have too many distinctive values. For `education`, we have

```
. * Tabulate all values taken by a single variable
. tabulate education
```

Completed education	Freq.	Percent	Cum.
0	82	1.91	1.91
1	7	0.16	2.07
2	20	0.47	2.54
3	32	0.75	3.29
4	26	0.61	3.89
5	30	0.70	4.59
6	123	2.87	7.46
7	35	0.82	8.28
8	78	1.82	10.09
9	117	2.73	12.82
10	167	3.89	16.71
11	217	5.06	21.77
12	1,510	35.20	56.97
13	263	6.13	63.10
14	432	10.07	73.17
15	172	4.01	77.18
16	535	12.47	89.65
17	317	7.39	97.04
99	127	2.96	100.00
Total	4,290	100.00	

Note that the variable label rather than the variable name is used as a header. The values are generally plausible, with 35% of the sample having a highest grade completed of exactly 12 years (high school graduate). The 7% of observations with 17 years most likely indicates a postgraduate degree (a college degree is only 16 years). The value 99 for 3% of the sample most likely is a missing-data code. Surprisingly, 2% appear to have completed no years of schooling. As we explain next, these are also observations with missing data.

#### 2.4.4 Using original documentation

At this stage, it is really necessary to go to the original documentation.

The `mus202psid92mcb.pdf` file, generated as part of the data extraction from the PSID website, states that for the `er30748` variable a value of 0 means “inappropriate” for various reasons given in the codebook; the values 1–16 are the highest grade or year of school completed; 17 is at least some graduate work; and 99 denotes not applicable or did not know.

Clearly, the `education` values of both 0 and 99 denote missing values. Without using the codebook, we may have misinterpreted the value of 0 as meaning 0 years of schooling.

#### 2.4.5 Missing values

It is best at this stage to flag missing values and to keep all observations rather than to immediately drop observations with missing data. In later analysis, only those observations with data missing on variables essential to the analysis need to be dropped. The characteristics of individuals with missing data can be compared with those having complete data. Data with a missing value are recoded with a missing-value code.

For `education`, the missing-data values 0 or 99 are replaced by . (a period), which is the default Stata missing-value code. Rather than create a new variable, we modify the current variable by using `replace`, as follows:

```
. * Replace missing values with missing-data code  
. replace education = . if education == 0 | education == 99  
(209 real changes made, 209 to missing)
```

Using the double equality and the symbol `|` for the logical operator `or` is detailed in section [1.3.6](#). As an example of the results, we list observations 46–48:

```
. * Listing of variable including missing value  
. list education in 46/48, clean  
  
      educat~n  
46.          12  
47.          .  
48.          16
```

Evidently, the original data on `education` for the 47th observation equaled 0 or 99. This has been changed to missing.

Subsequent commands using the `education` variable will drop observations with missing values. For example,

```
. * Example of data analysis with some missing values  
. summarize education age
```

Variable	Obs	Mean	Std. dev.	Min	Max
education	4,081	12.5533	2.963696	1	17
age	4,290	38.37995	5.650311	30	50

For `education`, only the 4,081 nonmissing values are used, whereas for `age`, all 4,290 of the original observations are available.

If desired, you can use more than one missing-value code. This can be useful if you want to keep track of reasons why a variable is missing. The extended missing codes are `.a`, `.b`, ..., `.z`. For example, we could instead have typed

```
. * Assign more than one missing code  
. replace education = .a if education == 0  
. replace education = .b if education == 99
```

When we want to apply multiple missing codes to a variable, it is more convenient to use the `mvdecode` command, which is similar to the `recode` command (discussed in section [2.4.7](#)), which changes variable values or ranges of values into missing-value codes. The reverse command, `mvencode`, changes missing values to numeric values.

Care is needed when missing values are used. In particular, missing values are treated as large numbers, higher than any other number. The ordering is that all numbers are less than `.`, which is less than `.a`, and so on. The command

```
. * This command will include missing values  
. list education in 40/60 if education > 16, clean  
      educat~n  
45.        17  
47.        .  
60.        17
```

lists the missing value for observation 47 in addition to the two values of 17. If this is not desired, we should instead use

```

. * This command will not include missing values
. list education in 40/60 if education > 16 & !missing(education), clean
    educat~n
45.          17
60.          17

```

Now observation 47 with the missing observation has been excluded.

The issue of missing values also arises for `earnings` and `hours`. From the codebook, we see that a zero value may mean missing for various reasons, or it may be a true zero if the person did not work. True zeros are indicated by `er30749=0` or 2, but we did not extract this variable. Thus, it is not unusual to have to extract data several times. Rather than extract this additional variable, as a shortcut we note that `earnings` and `hours` are missing for the same reasons that `education` is missing. Thus,

```

. * Replace missing values with missing-data code
. replace earnings = . if missing(education)
(209 real changes made, 209 to missing)
. replace hours = . if missing(education)
(209 real changes made, 209 to missing)

```

## 2.4.6 Imputing missing data

The standard approach in microeconomics is to drop observations with missing values, called listwise deletion. The loss of observations generally leads to less precise estimation and inference. More importantly, it may lead to sample-selection bias in regression if the retained observations have unrepresentative values of the dependent variable conditional on regressors; see section 19.10.

An alternative to dropping observations is to impute missing values. The norm in microeconomics studies is to use only the original data. The `ipolate` command uses linear interpolation. The community-contributed `hotdeck` command ([Mander and Clayton 1999](#)) implements hotdeck imputation. A more promising approach, though one more advanced, is multiple imputation. This produces  $M$  different imputed datasets (for example,  $M = 20$ ), fits the model  $M$  times, and performs inference that allows for the uncertainty in both estimation and data imputation. Multiple implementation using the `mi impute` command is presented in section 30.5.

## 2.4.7 Transforming data (generate, replace, egen, recode)

After handling missing values, we have the following for the key variables:

```
. * Summarize cleaned-up data  
. summarize sex age education earnings
```

Variable	Obs	Mean	Std. dev.	Min	Max
sex	4,290	1	0	1	1
age	4,290	38.37995	5.650311	30	50
education	4,081	12.5533	2.963696	1	17
earnings	4,081	28706.65	32279.12	0	999999

We now turn to recoding existing variables and creating new variables. The basic commands are `generate` and `replace`. It can be more convenient, however, to use the additional commands `recode`, `egen`, and `tabulate`. These are often used in conjunction with the `if` qualifier and the `by` prefix. We present many examples throughout the book.

### The generate and replace commands

The `generate` command is used to create new variables, often using standard mathematical functions. The syntax of the command is

```
generate [ type ] newvar[ :lblname ] =exp [ if ] [ in ] [ , before(varname) after(varname) ]
```

where for numeric data the default type is `float`, but this can be changed, for example, to `double`.

It is good practice to assign a unique identifier to each observation if one does not already exist. A natural choice is to use the current observation number stored as the system variable `_n`.

```
. * Create identifier using generate command  
. generate id = _n
```

We use this identifier for simplicity, though for these data the `er30001` and `er30002` variables when combined provide a unique PSID identifier.

The following command creates a new variable for the natural logarithm of earnings:

```
. * Create new variable using generate command  
. generate lnearns = ln(earnings)  
(498 missing values generated)
```

Missing values for `ln(earnings)` are generated whenever `earnings` data are missing. Additionally, missing values arise when `earnings`  $\leq 0$  because it is then not possible to take on the logarithm.

The `replace` command is used to replace some or all values of an existing variable. We already illustrated this when we created missing-values codes.

### **The egen command**

The `egen` command is an extension to `generate` that enables creation of variables that would be difficult to create using `generate`. For example, suppose we want to create a variable that for each observation equals sample average earnings provided that sample earnings are nonmissing. The command

```
. * Create new variable using egen command  
. egen aveearnings = mean(earnings) if !missing(earnings)  
(209 missing values generated)
```

creates a variable equal to the average of earnings for those observations not missing data on `earnings`.

The `egen` command supports many functions; see `help egen`. It is often used in conjunction with the `by` prefix.

### **The recode command**

The `recode` command is an extension to `replace` that recodes categorical variables and generates a new variable if the `generate()` option is used. The command

```
. * Replace existing data using the recode command  
. recode education (1/11=1) (12=2) (13/15=3) (16/17=4), generate(edcat)  
(4074 differences between education and edcat)
```

creates a new variable, `edcat`, that takes on a value of 1, 2, 3, or 4 corresponding to, respectively, less than high school graduate, high school graduate, some college, and college graduate or higher. The `edcat` variable is set to missing if `education` does not lie in any of the ranges given in the `recode` command.

### The decode command

The `decode` command converts numeric data to string data, where the string values are those given in a preexisting label.

The numeric variable `sex` has label `gender`. We create a string variable named `str_sex` as follows:

```
. * Convert numeric variable to string using decode and preexisting label
. list sex in 1, nolabel clean      // List the numeric value
      sex
1.    1
. list sex in 1, clean           // List the associated label
      sex
1.  male
. decode sex, generate(str_sex)   // Make a new string variable
. list str_sex in 1, clean       // List the new string variable
      str_sex
1.  male
```

The first two `list` commands show that the first observation of the numeric variable `sex` had value 1 and label value `male`. The `decode` command then created a string variable whose values are determined by whatever label has previously been assigned to `sex`. This yields a variable whose first observation is `male`.

### The encode command

The `encode` command converts string data to numeric data and creates an associated label whose values are those of the original string variable.

We convert the just created string variable `str_sex` to a numeric variable.

```

. * Convert string variable to numeric with label using encode
. encode str_sex, generate(num_sex) // Make a new numeric variable
. list num_sex in 1, nolabel clean // List the numeric value
      num_sex
1.          1
. list num_sex in 1, clean           // List the associated label
      num_sex
1.    male

```

The value label for the newly created numeric variable `num_sex` is also given the name `num_sex`.

### **The `by` prefix**

The `by varlist`: prefix repeats a command for each group of observations for which the variables in `varlist` are the same. The data must first be sorted by `varlist`. This can be done by using the `sort` command, which orders the observations in ascending order according to the variables given in the command.

The `sort` command and the `by` prefix are more compactly combined into the `bysort` prefix. For example, suppose we want to create for each individual a variable that equals the sample average earnings for all persons with that individual's years of education. Then we type

```

. * Create new variable using bysort prefix
. bysort education: egen aveearnsbyed = mean(earnings)
(209 missing values generated)
. sort id

```

The final command, one that returns the ordering of the observation to the original ordering, is not required. But it could make a difference in subsequent analysis if, for example, we were to work with a subsample of the first 1,000 observations.

### **Indicator variables**

Consider creating a variable indicating whether earnings are positive. While there are several ways to proceed, we describe only our recommended method.

The most direct way is to use `generate` with logical operators:

```
. * Create indicator variable using generate command with logical operators
. generate d1 = earnings > 0 if !missing(earnings)
(209 missing values generated)
```

The expression `d1 = earnings > 0` creates an indicator variable equal to 1 if the condition holds and 0 otherwise. Because missing values are treated as large numbers, we add the condition `if !missing(earnings)` so that in those cases `d1` is set equal to missing.

Using `summarize`, we obtain

```
. summarize d1
```

Variable	Obs	Mean	Std. dev.	Min	Max
d1	4,081	.929184	.2565486	0	1

We can see that about 93% of the individuals in this sample had some earnings in 1992. We can also see that we have  $0.929184 \times 4081 = 3792$  observations with a value of 1, 289 observations with a value of 0, and 209 missing observations.

### Set of indicator variables

A complete set of mutually exclusive categorical indicator dummy variables can be created in several ways.

For example, suppose we want to create mutually exclusive indicator variables for less than high school graduate, high school graduate, some college, and college graduate or more. The starting point is the `edcat` variable, created earlier, which takes on the values 1–4.

We can use `tabulate` with the `generate()` option.

```
. * Create a set of indicator variables using tabulate with generate() option
. qui tabulate edcat, generate(eddummy)
. summarize eddummy*
```

Variable	Obs	Mean	Std. dev.	Min	Max
eddummy1	4,081	.2087724	.4064812	0	1
eddummy2	4,081	.3700074	.4828655	0	1
eddummy3	4,081	.2124479	.4090902	0	1
eddummy4	4,081	.2087724	.4064812	0	1

The four means sum to one, as expected for four mutually exclusive categories. Note that if `edcat` had taken on values 4, 5, 7, and 9, rather than 1–4, it would still generate variables numbered `eddummy1`–`eddummy4`.

It is usually not necessary to actually create a set of indicator variables. Instead, we can include factor variables in the variable list; see section [1.3.4](#). For example,

```
. * Set of indicator variables using factor variables - no category is omitted
. summarize i.edcat
```

Variable	Obs	Mean	Std. dev.	Min	Max
edcat					
1	4,081	.2087724	.4064812	0	1
2	4,081	.3700074	.4828655	0	1
3	4,081	.2124479	.4090902	0	1
4	4,081	.2087724	.4064812	0	1

Almost all commands with a variable list permit use of factor variables in the variable list. Exceptions include a few estimation commands such as the `exlogistic` command. In such cases, the older `xi` prefix can be used instead. For details, type `help xi`.

## Interactions

Interactive variables can be created in the obvious manner. For example, to create an interaction between the binary earnings indicator `d1` and the continuous variable `education`, type

```
. * Create interactive variable using generate commands
. generate d1education = d1*education
(209 missing values generated)
```

Rather than create the interactive variables, we can use factor-variable operators. For example,

- . \* Set of interactions using factor-variable operators
- . summarize i.edcat#c.earnings

Variable	Obs	Mean	Std. dev.	Min	Max
edcat# c.earnings					
1	4,081	3146.368	8286.325	0	80000
2	4,081	8757.823	15710.76	0	215000
3	4,081	6419.347	16453.14	0	270000
4	4,081	10383.11	32316.32	0	999999

Here the # operator is used to create interactions, the i. operator is applied to a categorical variable, and the c. operator is for a continuous variable.

We can similarly use factor-variable operators to create interactions between categorical variables and to create interactions between continuous variables; see section [1.3.4](#).

Factor-variable operators enable one to obtain marginal effects in regression models with interactions using the `margins` command.

### Demeaning

Suppose we want to include a quadratic in age as a regressor. The marginal effect of age is much easier to interpret if we use the demeaned variables  $(age - \bar{age})$  and  $(age - \bar{age})^2$  as regressors.

- . \* Create demeaned variables
- . egen double aveage = mean(age)
- . generate double agedemean = age - aveage
- . generate double agesqdemmean = agedemean^2
- . summarize agedemean agesqdemmean

Variable	Obs	Mean	Std. dev.	Min	Max
agedemean	4,290	2.32e-15	5.650311	-8.379953	11.62005
agesqdemmean	4,290	31.91857	32.53392	.1443646	135.0255

We expect the `agedemean` variable to have an average of zero. We specified `double` to obtain additional precision in the floating-point calculations. In the case at hand, the mean of `agedemean` is on the order of  $10^{-15}$  instead of  $10^{-6}$ , which is what single-precision calculations would yield.

## 2.4.8 Saving and exporting data

At this stage, the dataset may be ready for saving. For illustrative purposes, we will name the dataset `cleaneddata.dta`. Apart from a few variable labels, it is identical to `mus202psid92m.dta`, which is used in the book from section [2.5](#) on.

The `save` command creates a Stata data file. For example,

```
. * Save as Stata data file (also used in semiparametric regression chapter)
. save cleaneddata, replace
(file cleaneddata.dta not found)
file cleaneddata.dta saved
```

The `replace` option means that an existing dataset with the same name, if it exists, will be overwritten. The `.dta` extension is unnecessary because it is the default extension.

Stata version 17 datasets can be read using version 16 or version 15. They can also be read by version 14, provided there are fewer than 32,768 variables, but cannot be read by earlier versions of Stata. The `saveold` command saves a data file that can be read by even earlier versions of Stata. The `saveold` command can save for versions 11–13. For example,

```
. * Save as Stata data file readable by versions 12 and above
. saveold cleaneddata, version(12) replace
(saving in Stata 12 format, which can be read by Stata 11 or 12)
file cleaneddata.dta saved
```

The output indicates that this dataset can actually be read by versions 11 and above.

The data can also be saved in another format that can be read by programs other than Stata; see section [2.3.10](#) for a list of formats.

The `export delimited` command allows saving data as a delimited text file in a spreadsheet format. The default is a comma-separated file.

```
. * Save as comma-separated values spreadsheet  
. export delimited age education eddummy* earnings d1 hours  
>     using cleaneddata.csv, replace  
(file cleaneddata.csv not found)  
file cleaneddata.csv saved
```

The wildcard `*` in `eddummy` expands to create `eddummy1`–`eddummy4` per the rules for wildcards, given in section [1.3.5](#). The first two lines in `cleaneddata.csv` are

```
age,education,eddummy1,eddummy2,eddummy3,eddummy4,earnings,d1,hours  
40,9,1,0,0,0,22000,1,2340
```

A space-delimited formatted text file can also be created by using the `outfile` command:

```
. * Save as formatted text (ascii) file  
. outfile age education eddummy* earnings d1 hours using cleaneddata.asc, replace  
(file cleaneddata.asc not found)
```

The first line in `cleaneddata.asc` is then

40	9	1	0	0	0	22000
1		2340				

This file will take up a lot of space; less space is taken if the `comma` option is used. The `dictionary` option writes the file in Stata's dictionary format, which includes at the start of the file a description of the formatting.

## 2.4.9 Selecting the sample

Most commands will automatically drop missing values in implementing a given command. We may want to drop additional observations, for example, to restrict analysis to a particular age group.

This can be done by adding an appropriate `if` qualifier after the command. For example, if we want to summarize data for only those individuals 35–44 years old, then we type

```
. * Select the sample used in a single command using the if qualifier
. summarize earnings lnearns if age >= 35 & age <= 44
```

Variable	Obs	Mean	Std. dev.	Min	Max
earnings	2,114	30131.05	37660.11	0	999999
lnearns	1,983	10.04658	.9001594	4.787492	13.81551

Different samples are being used here for the two variables because for the 131 observations with 0 earnings, we have data on `earnings` but not on `lnearns`. The `if` qualifier uses logical operators, defined in section [1.3.6](#).

However, for most purposes, we would want to use a consistent sample. For example, if separate earnings regressions were run in levels and in logs, we would usually want to use the same sample in the two regressions.

The `drop` and `keep` commands allow sample selection for the rest of the analysis. The `keep` command explicitly selects the subsample to be retained. Alternatively, we can use the `drop` command, in which case the subsample retained is the portion not dropped. The sample dropped or kept can be determined by using an `if` qualifier or a variable list or by defining a range of observations.

For the current example, we use

```
. * Select the sample using command keep
. keep if (!missing(lnearns)) & (age >= 35 & age <= 44)
(2,307 observations deleted)
. summarize earnings lnearns
```

Variable	Obs	Mean	Std. dev.	Min	Max
earnings	1,983	32121.55	38053.31	120	999999
lnearns	1,983	10.04658	.9001594	4.787492	13.81551

This command keeps the data provided: `lnearns` is nonmissing and  $35 \leq \text{age} \leq 44$ . Note that now `earnings` and `lnearns` are summarized for the same 1,983 observations.

As a second example, the commands

```
. * Select the sample using keep and drop commands
. use cleaneddata, clear
. keep lnearns age
. drop in 1/1000
(1,000 observations deleted)
```

will lead to a sample that contains data on all but the first 1,000 observations for just the two variables `lnearns` and `age`. The `use cleaneddata` command is added because the previous example had already dropped some of the data.

If we want to speed up analysis of large datasets, especially exploratory analysis, it can be useful to work with a random subset of the data. The command `keep if runiform() < 0.1`, for example, will keep a subsample of approximately 10% of the observations because the `runiform()` command provides pseudorandom draws of the uniform distribution on (0, 1). Alternatively, the command `sample 10`, for example, will draw a sample of exactly 10% of the observations; the option `count` instead enables drawing a sample with a specified number of observations. In all cases, one should first use the `set seed` command to ensure reproducibility.

## 2.4.10 Time-series data

For time-series data that include a time variable, the `tsset` command declares the data to be a time series, allowing the use of Stata's time-series operators.

For example, suppose data are monthly and the variable `month` is an integer that increases by one for each month. Then the command `tsset month` declares the data series to be a time series with time variable `month`.

It is then possible to use time-series operators, such as lags, differences, and leads. For example, the command `generate ylag = 1.y` creates a variable that equals variable `y` in the preceding month. Similarly, `112.y` denotes the 12th lag of `y`, or the value in the same month in the previous year, and `d.y` denotes the first difference or monthly change in variable `y`. If data are unavailable, then a missing value is generated. For example, for the first observation, it is not possible to create `1.y`.

A major complication is defining a time variable in a format that Stata recognizes because time-series datasets often store the date variable as a string variable. For example, suppose we have monthly data with variable `month` stored as “February 1, 1960” or as “2/1/1960”. This can be converted to a number using the `date()` function. In this example, we use `generate date2 = date(month, "MDY")` because the date string variable was ordered month, day, year. Stata normalizes dates as starting at 1/1/1960, so, for example, 2/1/1960 becomes 31. Because we have monthly data, we convert this to the number of months since 1960 using command `generate date3 = mofd(date2)`. Variable `date3` can be used immediately in a `tsset` command, but for proper dates to appear on graphs, we should give a date format. Here `date3` is months since 1960, so we give command `format %tm date3` because `%tm` is the format for monthly data.

The particulars in the preceding example will change according to whether data are daily, weekly, monthly, quarterly, yearly, ... and the exact way that dates appear in the original data. For details, see [M-5] **date()** and [D] **Datetime**.

## 2.5 Manipulating datasets

Useful manipulations of datasets include reordering observations or variables, temporarily changing the dataset but then returning to the original dataset, breaking one observation into several observations (and vice versa), and combining more than one dataset. We use `mus202psid92m.dta` obtained from PSID data on men aged 30–50 years in 1992. The dataset is identical to file `cleaneddata.dta` created by the preceding code, but with additional edits to variable labels.

### 2.5.1 Ordering observations and variables

Some commands, such as those using the `by` prefix, require sorted observations. The `sort` command orders observations in ascending order according to the variables in the command. The `sort, stable` command ensures that the previous ordering of tied values of the sort variables is maintained. The `gsort` command allows ordering to be in descending order.

You can also reorder the variables by using the `order` command. This can be useful if, for example, you want to distribute a dataset to others with the most important variables appearing as the first variables in the dataset.

### 2.5.2 Preserving and restoring a dataset

In some cases, it is desirable to temporarily change the dataset, perform some calculation, and then return the dataset to its original form. An example involving the computation of marginal effects is presented in section [13.5.4](#). The `preserve` command preserves the data, and the `restore` command restores the data to the form they had immediately before `preserve`.

```

. * Commands preserve and restore illustrated
. use mus202psid92m, clear
. list age in 1/1, noheader clean
  1.    40
. preserve
. replace age = age + 1000
variable age was byte now int
(4,290 real changes made)
. list age in 1/1, noheader clean
  1.  1040
. restore
. list age in 1/1, noheader clean
  1.    40

```

As desired, the data have been returned to original values.

### 2.5.3 Data frames

The `preserve` and `restore` commands essentially allow use of two datasets rather than one. The `frame` commands, introduced in Stata 16, enable switching between multiple datasets.

One way to create a new data frame is to make a copy of an existing data frame. The following code first determines that the current data in memory have frame title `default`, renames this as `first`, and copies `first` to `second`.

```

. * Create a new data frame
. frame
  (current frame is default)
. frame rename default first
. frame copy first second

```

Next, we change the current frame to `second` and change variable `age`. One feature of data frames is the `frame` prefix, which enables one to execute a command on data in a specified frame. In this case, while in frame `second`, we list an observation in frame `first`.

```
. * Change to the new data frame and manipulate
. frame change second
. replace age = age + 1000
variable age was byte now int
(4,290 real changes made)
. list age in 1/1, noheader clean
1.    1040
. frame first: list age in 1/1, noheader clean
1.    40
```

We then return to the original dataset.

```
. * Revert back to the original data frame
. frame change first
. list age in 1/1, noheader clean
1.    40
```

For the examples in this book, it is sufficient to use the `preserve` and `restore` commands, so we do not make use of the `frame` commands. Nonetheless, the ability to switch between data frames is a great enhancement to Stata's data management capabilities that will be especially useful in preparing data ahead of implementation of the statistical methods that are the focus of this book.

#### 2.5.4 Collapsing and expanding datasets

The `collapse` command collapses the dataset to a dataset of summary statistics.

For example, we can create a dataset of the median values of `earnings` and `age` for each of the four education categories.

```

. * Collapse to dataset of medians of earnings and age for each value of edcat
. preserve
. keep if !missing(earnings) & !missing(age)
(209 observations deleted)
. collapse (median) earnings age, by(edcat)
. list, clean
    edcat    earnings    age
  1.        1      14000    38
  2.        2      22000    37
  3.        3      28000    38
  4.        4     41392.5   39
. restore

```

If the missing observations had not been dropped, then a fifth observation would have been created because of the missing values of `education`.

The `expand` command creates duplicate observations. The number of duplicates can be a fixed number, such as 3, or be determined by a variable. For example, we expand the immediately preceding dataset of four observations, with duplicates determined by the value of `edcat`, provided `edcat` is less than four.

```

. * Expand dataset with number of duplicate observations = edcat if edcat < 4
. preserve
. qui keep if !missing(earnings) & !missing(age)
. qui collapse (median) earnings age, by(edcat)
. expand edcat if edcat < 4
(3 observations created)

. list, clean
    edcat    earnings    age
  1.        1      14000    38
  2.        2      22000    37
  3.        3      28000    38
  4.        4     41392.5   39
  5.        2      22000    37
  6.        3      28000    38
  7.        3      28000    38
. restore

```

The `expandcl` command duplicates clusters.

## 2.5.5 Wide and long forms for a dataset

Some datasets may combine several observations into a single observation. For example, a single household observation may contain data for several household members, or a single individual observation may have data for each of several years. This format for data is called wide form. If instead these data are broken out so that an observation is for a distinct household member, or for a distinct individual–year pair, the data are said to be in long form.

The `reshape` command is detailed in section [8.10](#). It converts data from wide form to long form and vice versa. This is necessary if an estimation command requires data to be in long form, say, but the original dataset is in wide form. The distinction is important especially for analysis of panel data and multinomial data.

## 2.5.6 Merging datasets

The `merge` command combines two datasets to create a wider dataset; that is, new variables from the second dataset are added to existing variables of the first dataset. Common examples are data on the same individuals obtained from two separate sources that then need to be combined and data on supplementary variables or additional years of data.

Merging two datasets involves adding information from a dataset on disk to a dataset in memory. The dataset in memory is known as the master dataset.

Merging two datasets is straightforward if the datasets have the same number of observations and the merge is a line-to-line merge. Then line 10, for example, of one dataset is combined with line 10 of the other dataset to create a longer line 10. The `merge 1:1 _n` command performs this merge.

We consider instead a match-merge, where observations in the two datasets are combined if they have the same values for one or more identifying variables that are used to determine the match. In either case, when a match is made if a variable appears in both datasets, the default is for the master dataset value to be retained unless it is missing, in which case it is replaced by the value in the second dataset. If a variable exists only in the second dataset, then it is added as a variable to the master dataset.

To demonstrate a match-merge, we create two datasets from the dataset used in this chapter. The first dataset comprises every third observation with data on `id`, `education`, and `earnings`:

```
. * Create first dataset with every third observation
. use mus202psid92m, clear
. keep if mod(_n,3) == 0
(2,860 observations deleted)
. keep id education earnings
. list in 1/4, clean
      educat~n    earnings    id
    1.          16     38708     3
    2.          12     3265      6
    3.          11    19426      9
    4.          11    30000     12
. qui save merge1, replace
```

The `keep if mod(_n,3) == 0` command keeps an observation if the observation number (`_n`) is exactly divisible by 3, so every third observation is kept. Because `id = xn` for these data, by saving every third observation, we are saving observations with `id` equal to 3, 6, 9, ....

The second dataset comprises every second observation with data on `id`, `education`, and `hours`:

```
. * Create second dataset with every second observation
. use mus202psid92m, clear
. keep if mod(_n,2) == 0
(2,145 observations deleted)
. keep id education hours
. list in 1/4, clean
      educat~n    hours    id
    1.          12   2008.00     2
    2.          12   2200.00     4
    3.          12    552.00     6
    4.          17   3750.00     8
. qui save merge2, replace
```

Now, we are saving observations with `id` equal to 2, 4, 6, ....

Now, we merge the two datasets by using the `merge 1:1` command. This requires that the identifying variable or variables for the match uniquely identify each observation in each dataset.

In our case, the datasets differ in both the observations included and the variables included, though there is considerable overlap. We perform a 1:1 match-merge on `id` to obtain

```
. * Merge 1:1 two datasets with some observations and variables different
. clear
. use merge1
. sort id
. merge 1:1 id using merge2
```

Result	Number of obs
Not matched	2,145
from master	715   (_merge==1)
from using	1,430   (_merge==2)
Matched	715   (_merge==3)

```
. sort id
. list in 1/4, clean
      educat~n    earnings     id      hours      _merge
1.        12          .     2    2008.00    Using only (2)
2.        16      38708     3          .    Master only (1)
3.        12          .     4    2200.00    Using only (2)
4.        12      3265     6     552.00     Matched (3)
```

Recall that observations from the master dataset have `id` equal to 3, 6, 9, ... and observations from the second dataset have `id` equal to 2, 4, 6, .... Data for `education` and `earnings` are always available because they are in the master dataset. But observations for `hours` come from the second dataset; they are available when `id` is 2, 4, 6, ... and are missing otherwise.

The same result is obtained if the roles of `merge1.dta` and `merge2.dta` are reversed. The `sort` commands are unnecessary; they simply ensure that observations in the merged dataset are ordered by `id`. For simplicity, we matched on just one variable, `id`, but one can match on several variables.

The `merge` command creates a variable, `_merge`, that takes on a value of 1 if the variables for an observation all come from the master dataset, a value

of 2 if they all come from only the second dataset, and a value of 3 if for an observation some variables come from the master and some from the second dataset. After using `merge`, you should check that the number of observations for each value of `_merge` matches your expectations.

There are several options when using `merge`. The `update` option varies the action `merge` takes when an observation is matched. By default, the master dataset is held inviolate—if `update` is specified, values from the master dataset are retained if the same variables are found in both datasets. However, the values from the merging dataset are used in cases where the variable is missing in the master dataset. The `replace` option, allowed only with the `update` option, specifies that even if the master dataset contains nonmissing values, they are to be replaced with corresponding values from the merging dataset when corresponding values are not equal. A nonmissing value, however, will never be replaced with a missing value.

A common type of merge that arises is when one dataset has one observation per `id`, where `id` is the key variable for the match, and the other dataset can have multiple observations per `id`. This merge can be done using either an appropriate one-to-many match or an appropriate many-to-one match.

Suppose `mergea.dta` has observations uniquely identified by `id`, while `mergeb.dta` can have multiple observations per `id`. Then we can give commands

```
. use mergea  
. merge 1:m id using mergeb
```

where `id` needs to uniquely identify observations in the master file. The same merged dataset can be obtained using the commands

```
. use mergeb  
. merge m:1 id using mergea
```

where now `id` needs to uniquely identify observations in the using file.

## 2.5.7 Appending datasets

The `append` command creates a longer dataset, with the observations from the second dataset appended after all the observations from the first dataset. If the same variable has different names in the two datasets, the variable name in one of the datasets should be changed by using the `rename` command so that the names match.

```
. * Append two datasets with some observations and variables different
. clear
. use merge1
. append using merge2
. sort id
. list in 1/5, clean
      educat~n    earnings     id      hours
      1.          12          .      2    2008.00
      2.          16        38708      3          .
      3.          12          .      4    2200.00
      4.          12          .      6     552.00
      5.          12        3265      6          .
```

Now, `merge2.dta` is appended to the end of `merge1.dta`. The combined dataset has observations 3, 6, 9, ..., 4290 followed by observations 2, 4, 6, ..., 4290. We then sort on `id`. Now both every second and every third observation is included, so after sorting, we have observations 2, 3, 4, 6, 8, 9, .... Note, however, that no attempt has been made to merge the datasets. In particular, for the observation with `id = 6`, the `hours` variable is missing in observation 4, and the `earnings` variable is missing in observation 5. This is because the `hours` variable is missing from the master dataset and the `earnings` variable is missing from the using dataset. There was no attempt to merge the data.

In this example, to take full advantage of the data, we would need to merge the two datasets using the first dataset as the master, merge the two datasets using the second dataset as the master, and then append the two datasets.

## 2.6 Graphical display of data

Graphs visually demonstrate important features of the data. Different types of data require distinct graph formats to bring out these features. We emphasize methods for numerical data taking many values, particularly, nonparametric methods.

### 2.6.1 Stata graph commands

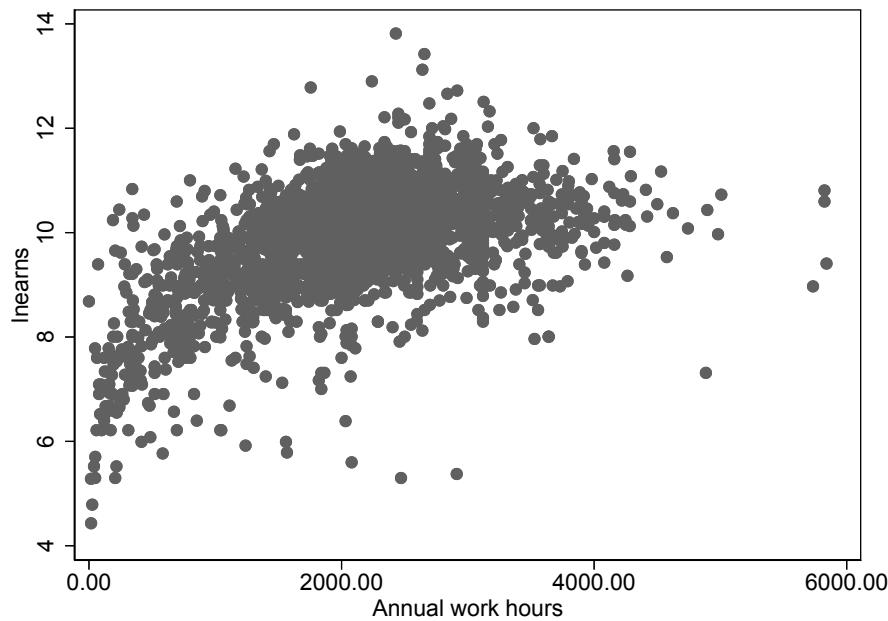
The Stata graph commands begin with the word `graph` (in some cases, this is optional) followed by the graph plottype, usually `twoway`. We cover several leading examples but ignore the plottypes `bar` and `pie` for categorical data.

#### Example graph commands

The basic graph commands are very short and simple to use. For example,

```
. use mus202psid92m, clear  
. twoway scatter lnearns hours
```

produces a scatterplot of `lnearns` on `hours`, shown in figure 2.1. Most graph commands support the `if` and `in` qualifiers, and some support weights.

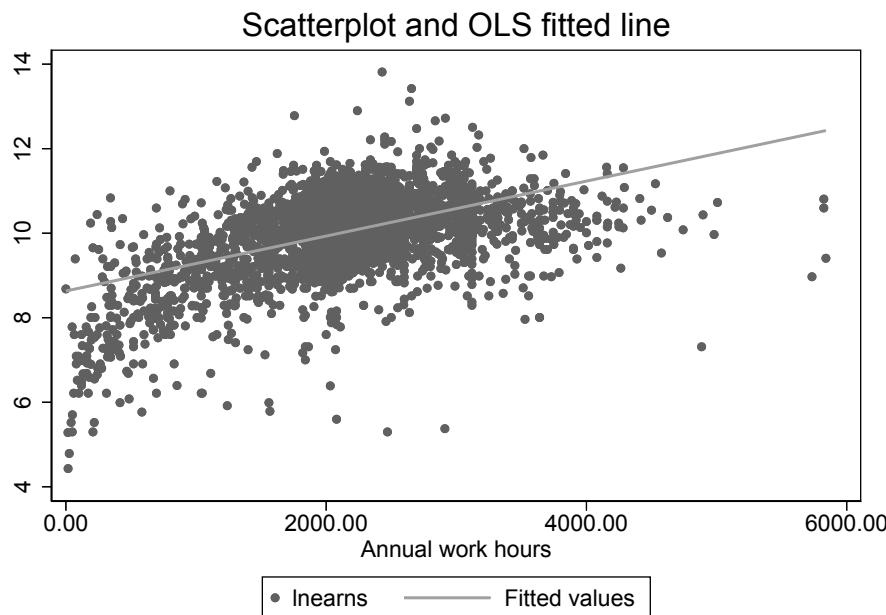


**Figure 2.1.** A basic scatterplot of log earnings on hours

In practice, however, customizing is often desirable. For example, we may want to display the relationship between `lnearns` and `hours` by showing both the data scatterplot and the ordinary least-squares (OLS) fitted line on the same graph. Additionally, we may want to change the size of the scatterplot data points, change the width of the regression line, and provide a title for the graph. We type

```
. * More advanced graphics command with two plots and with several options
. graph twoway (scatter lnearns hours, msizen(small))
>      (lfit lnearns hours, lwidth(medthick)),
>      title("Scatterplot and OLS fitted line")
```

The two separate components `scatter` and `lfit` are specified separately within parentheses. Each of these commands is given with one option, after the comma but within the relevant parentheses. The `msizen(small)` option makes the scatterplot dots smaller than the default, and the `lwidth(medthick)` option makes the OLS fitted line thicker than the default. The `title()` option for `twoway` appears after the last comma. The graph produced is shown in figure [2.2](#).



**Figure 2.2.** A more elaborate scatterplot of log earnings on hours

We often use lengthy graph commands that span multiple lines to produce template graphs that are better looking than those produced with default settings. In particular, these commands add titles and rescale the points, lines, and axes to a suitable size because the graphs printed in this book are printed in a much smaller space than a full-page graph in landscape mode. These templates can be modified for other applications by changing variable names and title text.

#### Saving and exporting graphs

Once a graph is created, it can be saved. Stata uses the term `save` to mean saving the graph in Stata's internal graph format, as a file with the `.gph` extension. This can be done by using the `saving()` option in a `graph` command or by typing `graph save` after the graph is created. When saved in this way, the graphs can be reaccessed and further manipulated at a later date.

Two or more Stata graphs can be combined into a single figure by using the `graph combine` command. For example, we save the first graph as `graph1.gph`, save the second graph as `graph2.gph`, and type the command

```
. * Combine graphs saved as graph1.gph and graph2.gph  
. graph combine graph1 graph2  
(output omitted)
```

Section [3.2.8](#) provides an example.

The Stata internal graph format (.gph) is not recognized by other programs, such as word processors. To save a graph in an external format, you would use the `graph export` command. For example,

```
. * Save graph as a scalable vector graphic  
. graph export mygraph.svg  
(output omitted)
```

Various formats are available, including PostScript (.ps), Encapsulated PostScript (.eps), scalable vector graphics (.svg), Windows Enhanced Metafile (.emf), GIF (.gif), JPEG (.jpg), PDF (.pdf), Portable Network Graphics (.png), and TIFF (.tif). The best format to select depends in part on what word processor is used; some trial and error may be needed.

#### Learning how to use graph commands

The Stata graph commands are extremely rich and provide an exceptional range of user control through a multitude of options.

A good way to learn the possibilities is to create a graph interactively in Stata. For example, from the menus, select **Graphics > Twoway graph (scatter, line, etc.)**. In the **Plots** tab of the resulting dialog box, select **Create...**, choose `Scatter`, provide a *Y variable* and an *X variable*, and then click on **Marker properties**. From the *Symbol* drop-down list, change the default to, say, `Triangle`. Similarly, cycle through the other options, and change the default settings to something else.

Once an initial graph is created, the point-and-click Stata Graph Editor allows further customizing of the graph, such as adding text and arrows wherever desired. This is an exceptionally powerful tool that we do not pursue here; for a summary, see [G-1] **Graph Editor**. The Graph Recorder can even save sequences of changes to apply to similar graphs created from different samples.

Even given familiarity with Stata's graph commands, you may need to tweak a graph considerably to make it useful. For example, any graph that analyzes the `earnings` variable using all observations will run into problems because one observation has a large outlying value of \$999,999. Possibilities in that case are to drop outliers, plot with the `yscale(log)` option, or use log `earnings` instead.

We find it easiest to work with lengthy template graph commands, such as that given at the beginning of this subsection, and modify these as needed.

### 2.6.2 Box-and-whisker plot

The `graph box` command produces a box-and-whisker plot that is a graphical way to display data on a single series. The boxes cover the interquartile range, from the lower quartile to the upper quartile. The whiskers, denoted by horizontal lines, extend to cover most of or all the range of the data. Stata places the upper whisker at the upper quartile plus 1.5 times the interquartile range, or at the maximum of the data if this is smaller. Similarly, the lower whisker is the lower quartile minus 1.5 times the interquartile range, or the minimum should this be larger. Any data values outside the whiskers are represented with dots. Box-and-whisker plots can be especially useful for identifying outliers.

The essential command for a box-and-whisker plot of the `hours` variable is

```
. * Simple box-and-whisker plot  
. graph box hours  
(output omitted)
```

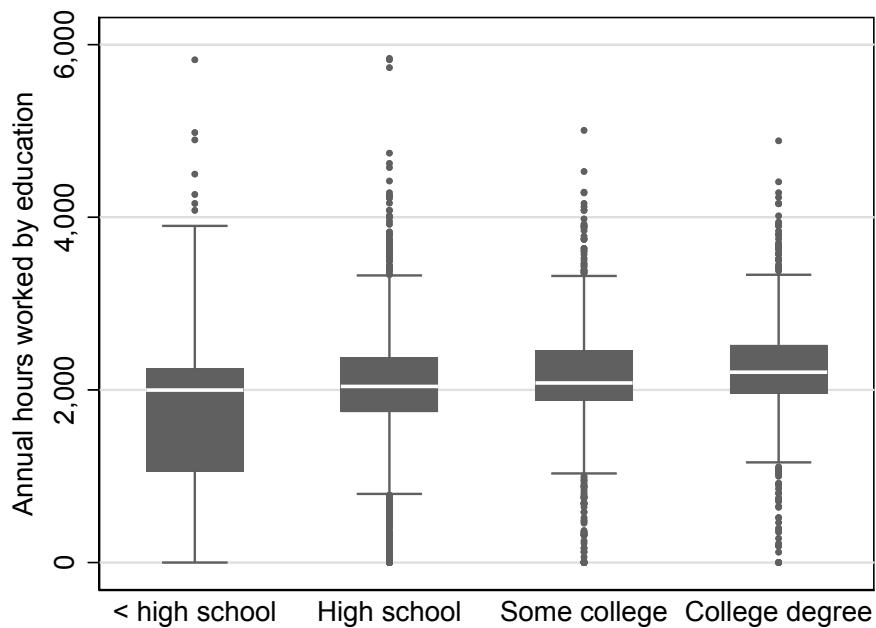
We want to present separate box plots of `hours` for each of four education groups by using the `over()` option. To make the plot more intelligible, we first provide labels for the four education categories as follows:

```
. use mus202psid92m, clear  
. label define edtype 1 "< high school" 2 "High school" 3 "Some college"  
> 4 "College degree"  
. label values edcat edtype
```

The `scale(1.2)` graph option is added for readability; it increases the size of text, markers, and line widths (by a multiple 1.2). The `marker()` option is added to reduce the size of quantities within the box, the `ytitle()` option is used to present the title, and the `yscale(titlegap(*5))` option is added to increase the gap between the *y*-axis title and the tick labels. We have

```
. * Box-and-whisker plot of single variable over several categories
. graph box hours, over(edcat) scale(1.2) marker(1,msize(vsmall))
>      ytitle("Annual hours worked by education") yscale(titlegap(*5))
```

The result is given in figure 2.3. The labels for `edcat`, rather than the values, are automatically given, making the graph much more readable. The filled-in boxes present the interquartile range, the intermediate line denotes the median, and data outside the whiskers appear as dots. For these data, annual hours are clearly lower for the lowest schooling group, and there are quite a few outliers. About 30 individuals appear to work in excess of 4,000 hours per year.



**Figure 2.3.** Box-and-whisker plots of annual hours for four categories of educational attainment

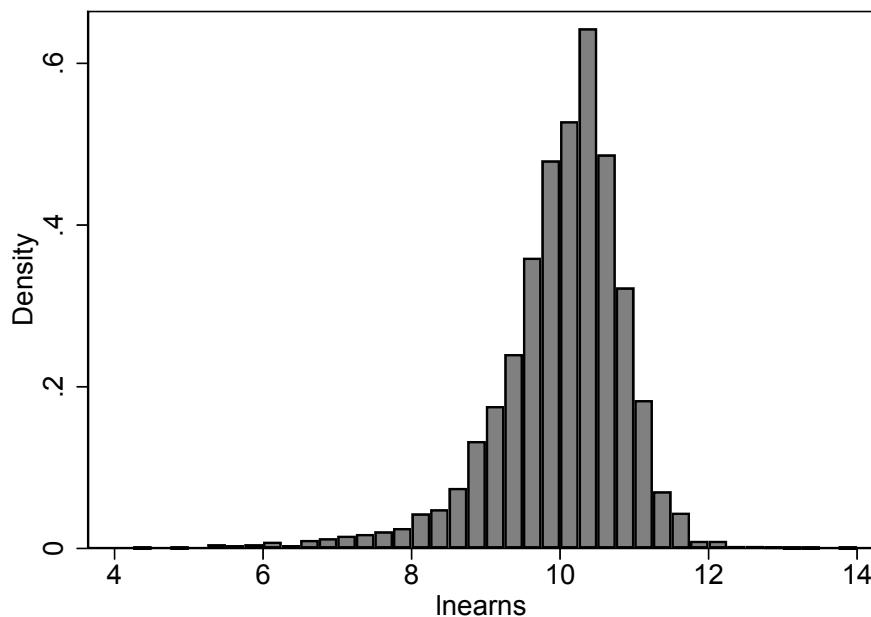
### 2.6.3 Histogram

The probability mass function or density function can be estimated using a histogram produced by the `histogram` command. The command can be used with `if` and `in` qualifiers and with weights. The key options are `width(#)` to set the bin width, `bin(#)` to set the number of bins, `start(#)` to set the lower limit of the first bin, and `discrete` to indicate that the data are discrete. The default number of bins is  $\min\{\sqrt{N}, 10 \times \ln N / \ln 10\}$ . Other options overlay a fitted normal density (the `normal` option) or a kernel density estimate (the `kdensity` option).

For discrete data taking relatively few values, there is usually no need to use the options.

For continuous data or for discrete data taking many values, it can be necessary to use options because the Stata defaults set bin widths that are not nicely rounded numbers and the number of bins might also not be desirable. For example, the output from `histogram lnearns` states that there are 35 bins, a bin width of 0.268, and a start value of 4.43. A better choice may be

```
. * Histogram with bin width and start value set
. histogram lnearns, width(0.25) start(4.0) scale(1.2)
(bin=40, start=4, width=.25)
```



## 2.6.4 Kernel density plot

For continuous data taking many values, a better alternative to the histogram is a kernel density plot. This provides a smoother version of the histogram in two ways: First, it directly connects the midpoints of the histogram rather than forming the histogram step function. Second, rather than giving each entry in a bin equal weight, it gives more weight to data that are closest to the point of evaluation.

Let  $f(x)$  denote the density. The kernel density estimate of  $f(x)$  at  $x = x_0$  is

$$\hat{f}(x_0) = \frac{1}{Nh} \sum_{i=1}^N K\left(\frac{x_i - x_0}{h}\right) \quad (2.1)$$

where  $K(\cdot)$  is a kernel function that places greater weight on points  $x_i$  close to  $x_0$ . More precisely,  $K(z)$  is symmetric around zero, integrates to one, and either  $K(z) = 0$  if  $|z| \geq z_0$  (for some  $z_0$ ) or  $K(z) \rightarrow 0$  as  $z \rightarrow \infty$ . A histogram with a bin width of  $2h$  evaluated at  $x_0$  can be shown to be the special case  $K(z) = 1/2$  if  $|z| < 1$ , and  $K(z) = 0$  otherwise.

A kernel density plot is obtained by choosing a kernel function,  $K(\cdot)$ ; choosing a width,  $h$ ; evaluating  $\hat{f}(x_0)$  at a range of values of  $x_0$ ; and plotting  $\hat{f}(x_0)$  against these  $x_0$  values.

The `kdensity` command produces a kernel density estimate. The command can be used with `if` and `in` qualifiers and with weights. The default window width or bandwidth is  $h = 0.9m/n^{1/5}$ , where  $m = \min(s_x, \text{iqr}_x/1.349)$  and  $\text{iqr}_x$  is the interquartile range of  $x$ . The `bwidth(#)` option allows a different width ( $h$ ) to be specified, with larger choices of  $h$  leading to smoother density plots. The `n(#)` option changes the number of evaluation points,  $x_0$ , from the default of  $\min(N, 50)$ . Other options overlay a fitted normal density (the `normal` option) or a fitted  $t$  density (the `student(#)` option).

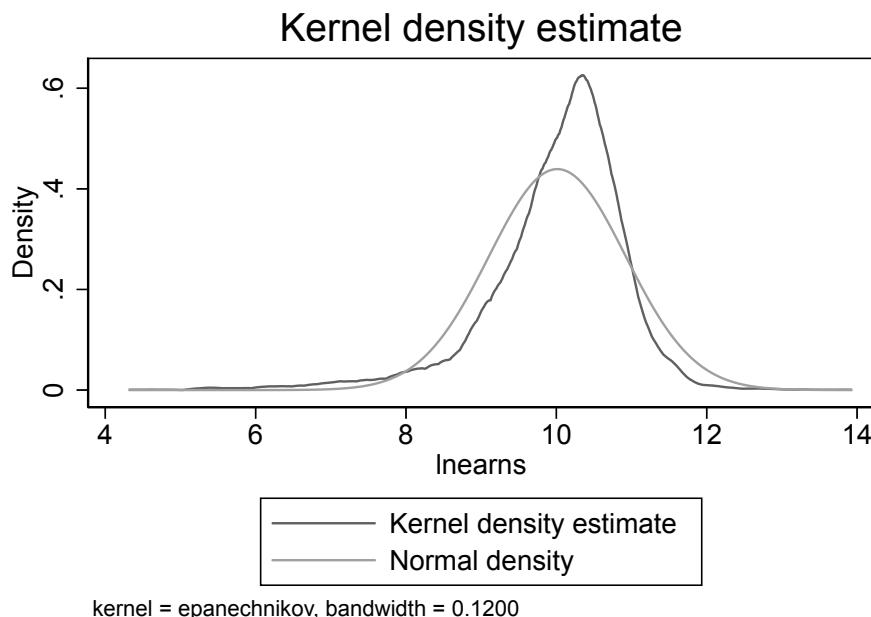
The default bandwidth is based on theory for when the underlying distribution is normal and the Gaussian kernel is used. Note that if the same default bandwidth is used, then the different kernels lead to different degrees of smoothing: 1) the `cosine` kernel gives the least smooth plot; 2) the `epan2`, `biweight`, `triangle`, `rectangle`, and `parzen` kernels give similar amounts of smoothing; and 3) the default `epanechnikov` and the `gaussian` kernels provide the smoothest plots.

The default kernel function is the Epanechnikov, which sets  $K(z) = (3/4)(1 - z^2/5)/\sqrt{5}$  if  $|z| < \sqrt{5}$ , and  $K(z) = 0$  otherwise. The `kernel()` option allows other kernels to be chosen. The `kernel(epan2)` option sets  $K(z) = (3/4)(1 - z^2)$  if  $|z| < 1$ , and  $K(z) = 0$  otherwise. The same results are obtained if the `epan2` bandwidth is  $\sqrt{5}$  times the default `epanechnikov` kernel. But if the same default bandwidth is used, then the two lead to different results. If a smoother kernel density plot is desired, then the default `epanechnikov` kernel with default bandwidth may be a good starting point. If a rougher plot is desired, then the `epan2` with the default bandwidth may be a better starting point.

From output not given, results from the command `kdensity lnearns` included a statement that the Epanechnikov kernel is used and the bandwidth equals 0.1227. To instead manually specify a bandwidth of 0.12, one overlaid by a fitted normal density, we type the command

```
. * Kernel density plot with bandwidth set and fitted normal density overlaid
. kdensity lnearns, bwidth(0.12) normal n(4000) scale(1.2)
```

which produces the graph in figure 2.5. This graph shows that the kernel density is more peaked than the normal and is somewhat left skewed.

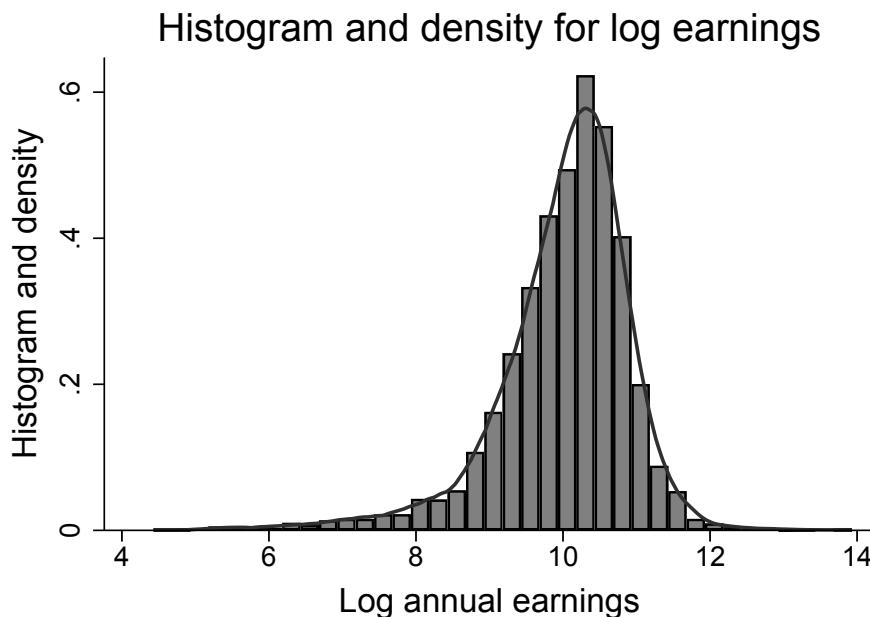


**Figure 2.5.** The estimated density of log earnings

The following code instead presents a histogram overlaid by a kernel density estimate. The histogram bin width is set to 0.25, the kernel density bandwidth is set to 0.2 using the `kdenopts()` option, and the kernel density plot line thickness is increased using the `lwidth(medthick)` option. Other options used here were explained in section [2.6.2](#). We have

```
. * Histogram and nonparametric kernel density estimate
. histogram lnearns if lnearns > 0, width(0.25) kdensity
>     kdenopts(bwidth(0.2) lwidth(medthick))
>     plotregion(style(none)) scale(1.2)
>     title("Histogram and density for log earnings")
>     xtitle("Log annual earnings", size(medlarge)) xscale(titlegap(*5))
>     ytitle("Histogram and density", size(medlarge)) yscale(titlegap(*5))
(bin=38, start=4.4308167, width=.25)
```

The result is given in figure [2.6](#). Both the histogram and the kernel density estimate indicate that the natural logarithm of earnings has a density that is mildly left skewed. A similar figure for the level of earnings is very right skewed.



**Figure 2.6.** Histogram and kernel density plot for log earnings

### 2.6.5 Twoway scatterplots and fitted lines

As we saw in figure 2.1, scatterplots provide a quick look at the relationship between two variables.

For scatterplots with discrete data that take on few values, it can be necessary to use the `jitter()` option. This option adds random noise so that points are not plotted on top of one another; see section 17.5.4 for an example.

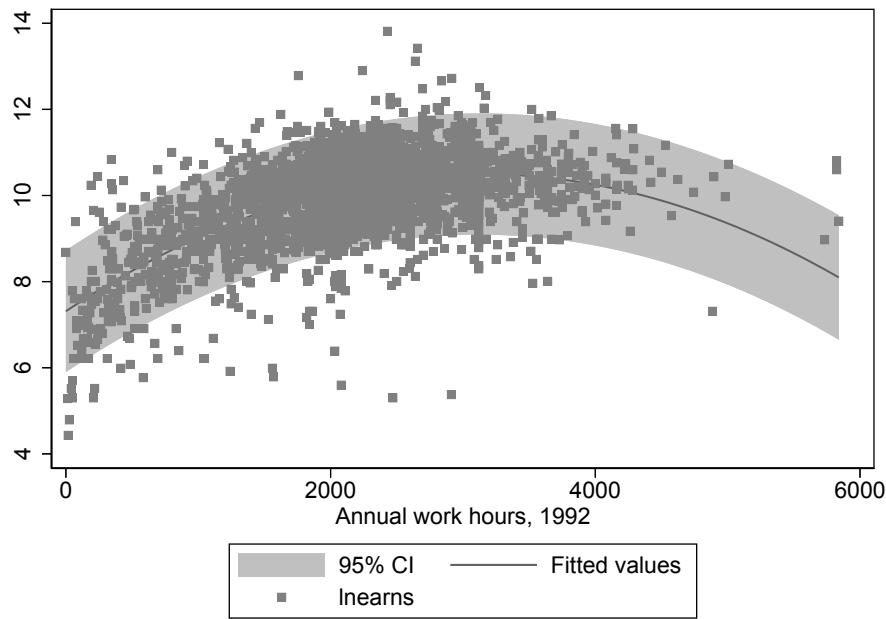
It can be useful to additionally provide a fitted curve. Stata provides several possibilities for estimating a global relationship between  $y$  against  $x$ , where by “global” we mean that a single relationship is estimated for all observations, and then for plotting the fitted values of  $y$  against  $x$ .

The `twoway lfit` command does so for a fitted OLS regression line, the `twoway qfit` command for a fitted quadratic regression curve, and the `twoway fpfit` command for a curve fit by fractional polynomial regression. The related `twoway` commands `lfitci`, `qfitci`, and `fpfitci` additionally provide confidence bands for predicting the conditional mean  $E(y|x)$  (by

using the `stdp` option) or for forecasting of the actual value of  $y|x$  (by using the `stdf` option).

For example, we may want to provide a scatterplot and fitted quadratic with confidence bands for the forecast value of  $y|x$  (the result is shown in figure 2.7):

```
. * Two-way scatterplot and quadratic regression curve with 95% ci for y|x
. twoway (qfitci lnearns hours, stdf) (scatter lnearns hours, msize(small))
```



**Figure 2.7.** Twoway scatterplot and fitted quadratic with confidence bands

### 2.6.6 Twoway scatterplots and locally fitted lines

An alternative curve-fitting approach to assuming linear or quadratic relationships between  $y$  and  $x$  is to use nonparametric methods. These fit a local relationship between  $y$  and  $x$ , at many values  $x_0$  of  $x$ . By “local”, we mean that prediction of  $y$  at each evaluation point  $x_0$  is based on weighted regression centered on data points in the neighborhood of  $x_0$ .

There are several nonparametric methods.

An easily understood example is a median-band plot. The range of  $x$  is broken into, say, 20 intervals; the medians of  $y$  and  $x$  in each interval are obtained; and the 20 medians of  $y$  are plotted against the 20 medians of  $x$ , with connecting lines between the points. The `twoway mband` command does this, and the related `twoway mspline` command uses a cubic spline to obtain a smoother version of the median-band plot.

### Local regression

We consider the regression model  $y = m(x) + u$ , where  $x$  is a scalar and the conditional mean function  $m(\cdot)$  is not specified. The goal is to estimate  $m(\cdot)$

A local regression estimate of  $m(x)$  at  $x = x_0$  is a local weighted average of  $y_i$ ,  $i = 1, \dots, N$ , that places great weight on observations for which  $x_i$  is close to  $x_0$  and little or no weight on observations for which  $x_i$  is far from  $x_0$ . Formally,

$$\hat{m}(x_0) = \sum_{i=1}^N w(x_i, x_0, h)y_i$$

where the weights  $w(x_i, x_0, h)$  sum over  $i$  to one and decrease as the distance between  $x_i$  and  $x_0$  increases. The weight depends on the closeness of  $x_i$  to  $x_0$  and on the parameter  $h$ , called a bandwidth parameter. Different methods use different weighting functions.

A plot is obtained by choosing a weighting function,  $w(x_i, x_0, h)$ ; choosing a bandwidth,  $h$ ; evaluating  $\hat{m}(x_0)$  at a range of values of  $x_0$ ; and plotting  $\hat{m}(x_0)$  against these  $x_0$  values. For example, provided  $N > 50$ , the Stata default for the `lpoly` command presented below is to evaluate at 50 evenly spaced data points between the minimum and maximum values of  $x$ . It may seem that there may be too few data points between each value of  $x_0$  to obtain a decent fit. This is avoided by using weights  $w(x_i, x_0, h)$  that average over much more than a small fraction of the data. Essentially, rolling windows are used to evaluate  $\hat{m}(x_0)$  at a range of values of  $x_0$ .

### Local constant and local linear kernel-weighted plots

Kernel regression at  $x = x_0$  uses the weight

$$w(x_i, x_0, h) = \frac{K\{(x_i - x_0)/h\}}{\sum_{j=1}^N K\{(x_j - x_0)/h\}}$$

where  $K(\cdot)$  is a kernel function defined after (2.1). For example, the `kernel(epan2)` option sets  $K(z) = (3/4)(1 - z^2)$  if  $|z| < 1$ , and  $K(z) = 0$  otherwise. Estimates depend crucially on the bandwidth chosen, with smaller  $h$  leading to more variable estimates of  $m(x_0)$ .

The kernel regression estimate at  $x = x_0$  can equivalently be obtained by minimizing

$$\sum_{i=1}^N w(x_i, x_0, h) \times (y_i - \alpha_0)^2$$

which is weighted regression on a constant where the kernel weights are largest for observations with  $x_i$  close to  $x_0$ . Then  $\hat{m}(x_0) = \hat{\alpha}_0$ . This estimator is also called the (kernel-weighted) local constant estimator.

The (kernel-weighted) local linear estimator of  $m(\cdot)$  additionally includes a slope coefficient and at  $x = x_0$  minimizes

$$\sum_{i=1}^N w(x_i, x_0, h) \times \{y_i - \alpha_0 - \beta_0(x_i - x_0)\}^2 \quad (2.2)$$

Again,  $\hat{m}(x_0) = \hat{\alpha}_0$ . This estimator has the advantage of better estimation of  $m(x_0)$  at values of  $x_0$  near the endpoints of the range of  $x$  because it allows for any trends near the endpoints.

More generally, the local polynomial estimator of degree  $p$  uses a polynomial of degree  $p$  in  $(x_i - x_0)$  in (2.2).

### The `lpoly` command

The `lpoly` command implements local polynomial estimation. The command has syntax

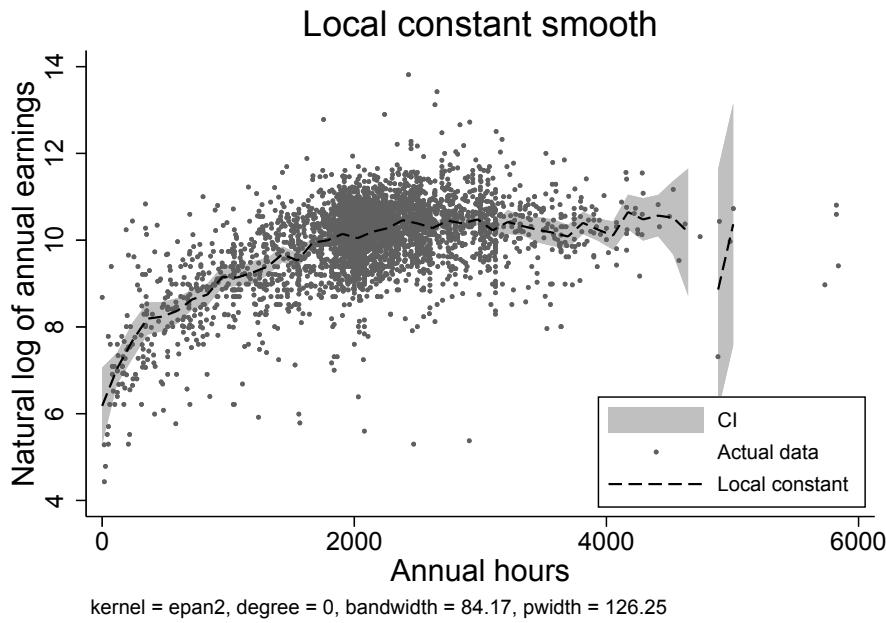
```
lpoly yvar xvar [if] [in] [weight] [, options]
```

The `degree(#)` option specifies the degree  $p$ , with local constant the default ( $p = 0$ ) and local linear ( $p = 1$ ) the most commonly used degrees. The `kernel()` option specifies the kernel (with default `epanechnikov`); the `bwidth(#)` option specifies the kernel bandwidth  $h$ ; and the `generate()` option saves the evaluation points  $x_0$  and estimates  $\hat{m}(x_0)$ . The default is to evaluate  $\hat{m}(x_0)$  at  $\min(N, 50)$  equally spaced values of  $x_0$  between the minimum and maximum values of  $x$ . The `at(varname)` option instead evaluates at the distinct values of `varname`, usually `at(x)`, where `x` is the regressor.

The default bandwidth is a plugin estimator of the optimal bandwidth assuming a constant bandwidth; see [R] **lpoly**. This default bandwidth is by no means perfect and in practice can undersmooth or oversmooth the data. The bandwidth then needs to be set by using the `bwidth()` option.

The following example illustrates the relationship between log earnings and hours worked. We present a local constant curve, using the `kernel(epan2)` option, and with 95% confidence bands added using the `ci` option. Additional options, explained below, modify the appearance of the graph.

```
. * Local constant with epan2 kernel and 95% confidence bands
. use mus202psid92m, clear
. lpoly lnearns hours, kernel(epan2) ci msize(tiny) lwidth(medthick)
> plotregion(style.none)) xtitle("Annual hours", size(medlarge))
> title("Local constant smooth") scale(1.1)
> ytitle("Natural log of annual earnings", size(medlarge))
> legend(pos(4) ring(0) col(1)) legend(size(small))
> legend(label(1 "CI") label(2 "Actual data") label(3 "Local constant"))
```



**Figure 2.8.** Local constant plot of log earnings against hours

The resulting graph is presented in figure 2.8. The confidence bands are much narrower in regions where there are many more observations on `hours` because then more observations are used in calculating the local average. For `hours` in excess of 4,500, there are generally too few observations to compute the local constant estimator because only observations within 84.17 hours (the bandwidth used) of the 50 equally spaced evaluation points are used in computation.

#### The `lowess` command

The locally weighted scatterplot smoothing estimator (`lowess`) is a variation of the local linear estimator that uses a variable bandwidth and tricubic kernel and downweights observations with large residuals (using a method that greatly increases the computational burden).

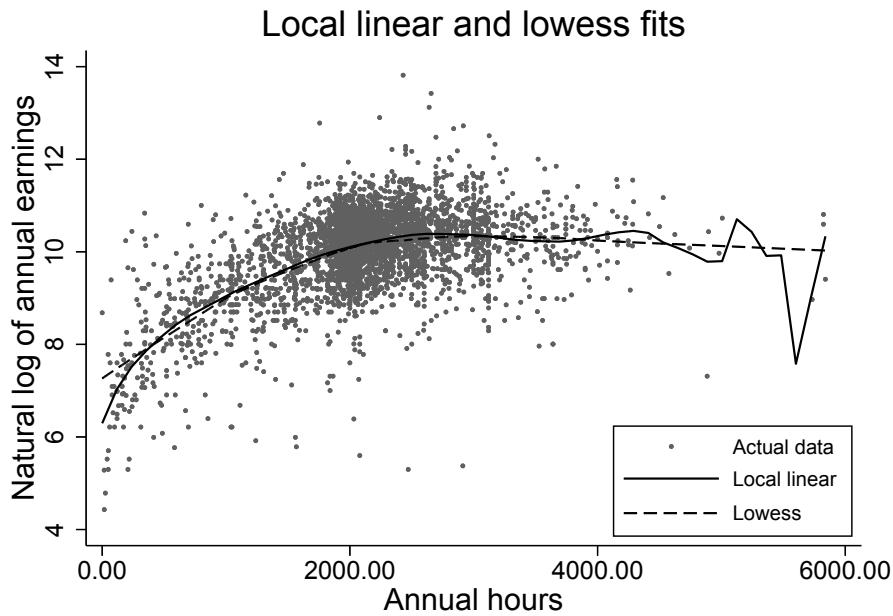
This estimator is obtained by using the `lowess` command. The bandwidth gives the fraction of the observations used to calculate  $\hat{m}(x_0)$  in the middle of the data, with a smaller fraction used toward the endpoints. The default value of 0.8 can be changed by using the `bwidth(#)` option, so as with the

local polynomial methods, a smoother plot is obtained by increasing the bandwidth.

We create a scatterplot (`scatter`) with a fitted lowess curve (`lowess`), along with a local linear curve (`lpoly`). The command is lengthy because of the detailed formatting commands used to produce a nicely labeled and formatted graph.

The `msize(tiny)` option is used to decrease the size of the dots in the scatterplot. The `lwidth(medthick)` option is used to increase the thickness of lines, and the `clstyle(p1)` option changes the style of the line for `lowess`. The `title()` option provides the overall title for the graph. The `xtitle()` and `ytitle()` options provide titles for the  $x$  axis and  $y$  axis, and the `size(medlarge)` option defines the size of the text for these titles. The `legend()` options place the graph legend at four o'clock (`pos(4)`) with text size `small` and provide the legend labels. We have

```
. * Scatterplot with lowess and local linear nonparametric regression
. graph twoway (scatter lnearns hours, msize(tiny))
>     (lpoly lnearns hours, kernel(epan2) degree(1) clstyle(p1) lwidth(thick)
>      bwidth(500)) (lowess lnearns hours, clstyle(p2) lwidth(thick)),
>      plotregion(style.none)) title("Local linear and lowess fits")
>      xtitle("Annual hours", size(medlarge)) scale(1.1)
>      ytitle("Natural log of annual earnings", size(medlarge))
>      legend(pos(4) ring(0) col(1)) legend(size(small))
>      legend(label(1 "Actual data") label(2 "Local linear") label(3 "Lowess"))
```



**Figure 2.9.** Local linear and lowess plots of log earnings against hours

The resulting graph is presented in figure 2.9. This command used the default bandwidth setting for `lowess` and greatly increased the `lpoly` bandwidth from its automatically selected value of 84.17 to 500. Even so, the local linear curve is too variable at high hours where the data are sparse. At low hours, however, the lowess estimator overpredicts, while the local linear estimator does not.

Figures 2.8 and 2.9 both indicate that log earnings increase with hours until about 2,500 hours and that a quadratic relationship may be appropriate.

In summary, the `lpoly` and `lowess` commands provide local regression curves that are especially useful when overlaid on a twoway scatterplot as in figure 2.9. For both commands, a larger bandwidth leads to a smoother curve. Both commands have a default plugin formula for determining the bandwidth, one that is printed on the resulting graph and stored as the scalar `r(bwidth)`. The user may want to change this value using the `bwidth()` option to obtain curves that are more or less smooth. For the `lpoly` command, different kernel functions can be specified using the `kernel()` option. The choice of kernel is of secondary importance, but note that a

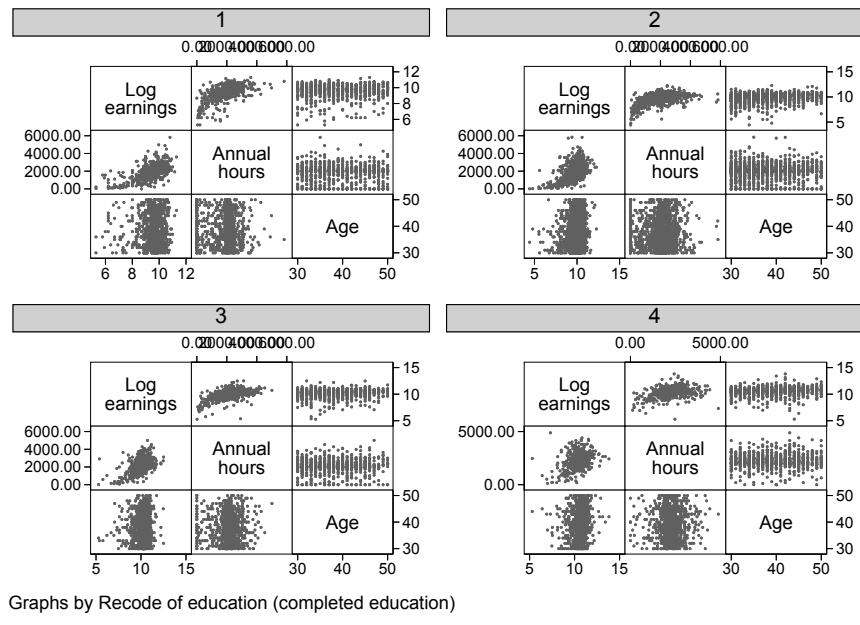
given bandwidth size, such as `bwidth(50)`, corresponds to different degrees of smoothness for different kernel functions, as already discussed for kernel density estimation in section [2.6.4](#).

The local constant and local linear plots from the `lpoly` command can also be obtained using the `nppgraph` command following the `npregress kernel` command. This is illustrated in section [14.6](#). The `npregress kernel` command is a richer command that is much more than a graphics command. It extends local constant and local linear regression to the case of multiple regressors and can be used, for example, to compute marginal effects for individual regressors in a model more flexible than the linear regression model; see chapter 27.

## 2.6.7 Multiple scatterplots

The `graph matrix` command provides separate bivariate scatterplots between several variables. Here we produce bivariate scatterplots (shown in figure [2.10](#)) of `lnearns`, `hours`, and `age` for each of the four education categories:

```
. * Multiple scatterplots
. label variable age "Age"
. label variable lnearns "Log earnings"
. label variable hours "Annual hours"
. graph matrix lnearns hours age, by(edcat) msize(tiny)
```



**Figure 2.10.** Multiple scatterplots of several variables for each level of education

Stata does not provide three-dimensional graphs, such as that for a nonparametric bivariate density estimate or for nonparametric regression of one variable on two other variables.

## 2.7 Additional resources

The key data management references are [U] *Stata User's Guide* and [D] *Stata Data Management Reference Manual*. Useful online `help` categories include 1) `double`, `string`, and `format` for data types; 2) `clear`, `use`, `import`, `infile`, and `outsheet` for data input; 3) `summarize`, `list`, `label`, `tabulate`, `generate`, `egen`, `keep`, `drop`, `recode`, `by`, `sort`, `merge`, `append`, and `collapse` for data management; and 4) `graph`, `graph box`, `histogram`, `kdensity`, `twoway`, `lpoly`, `lowess`, and `graph matrix` for graphical analysis.

The Stata graphics commands are quite flexible and Stata provides both an interactive Graph Editor and a Graph Recorder; see [G-1] **Graph Editor**. *A Visual Guide to Stata Graphics* by [Mitchell \(2022\)](#) provides many hundreds of template graphs with the underlying Stata code and an explanation for each.

## 2.8 Exercises

1. Type the command `display %10.5f 123.321`. Compare the results with those you obtain when you change the format `%10.5f` to, respectively, `%10.5e`, `%10.5g`, `%-10.5f`, and `%10,5f` and when you do not specify a format.
2. Consider the example of section [2.3](#) except with the variables reordered. Specifically, the variables are in the order `age`, `name`, `income`, and `female`. The three observations are 29 "Barry" 40.990 0; 30 "Carrie" 37.000 1; and 31 "Gary" 48.000 0. Use `input` to read these data, along with names, into Stata, and list the results. Use a text editor to create a comma-separated values file that includes variable names in the first line, read this file into Stata by using `import delimited`, and list the results. Then, drop the first line in the text file, read in the data by using `import delimited` with variable names assigned, and list the results. Finally, replace the commas in the text file with blanks, read the data in by using `infix`, and list the results.
3. Consider the dataset in section [2.4](#). The `er32049` variable is the last-known marital status. Rename this variable as `marstatus`, give the variable the label "marital status", and `tabulate marstatus`. From the codebook, marital status is married (1), never married (2), widowed (3), divorced or annulment (4), separated (5), not answered or do not know (8), and no marital history collected (9). Set `marstatus` to missing where appropriate. Use `label define` and `label values` to provide descriptions for the remaining categories, and `tabulate marstatus`. Create a binary indicator variable equal to 1 if the last-known marital status is married and equal to 0 otherwise, with appropriate handling of any missing data. Provide a summary of earnings by marital status. Create a set of indicator variables for marital status based on `marstatus`. Create a set of variables that interact these marital status indicators with earnings.
4. Consider the dataset in section [2.6](#). Create a box-and-whisker plot of `earnings` (in levels) for all the data and for each year of educational attainment (use variable `education`). Create a histogram of `earnings` (in levels) using 100 bins and a kernel density estimate. Do earnings in levels appear to be right skewed? Create a scatterplot of `earnings`

against `education`. Provide a single figure that uses `scatterplot`, `lfit`, and `lowess` of `earnings` against `education`. Add titles for the axes and graph heading.

5. Consider the dataset in section [2.6](#). Create kernel density plots for `lnearns` using the `kernel(epan2)` option with kernel  $K(z) = (3/4)(1 - z^2/5)$  for  $|z| < 1$  and using the `kernel(rectangle)` option with kernel  $K(z) = 1/2$  for  $|z| < 1$ . Repeat with the bandwidth increased from the default to 0.3. What makes a bigger difference, choice of kernel or choice of bandwidth? The comparison is easier if the four graphs are saved using the `saving()` option and then combined using the `graph combine` command.
6. Consider the dataset in section [2.6](#). For each of the available kernels that can be used with the `kdensity` command, obtain a kernel density plot for `lnearns` using the default bandwidth, and save the graph using the `saving()` option. Then, combine all graphs on one page using the `graph combine` command and options such as `rows(4) ysize(8) xsize(5)`. Comment on the relative smoothness of the various graphs.
7. Consider the dataset in section [2.6](#). Perform lowess regression of `lnearns` on `hours` using the default bandwidth and using bandwidth of 0.01. Does the bandwidth make a difference? A moving average of  $y$  after data are sorted by  $x$  is a simple case of nonparametric regression of  $y$  on  $x$ . Sort the data by `hours`. Create a centered 25-period moving average of `lnearns` with  $i$ th observation  $y_{ma_i} = 1/25 \sum_{j=-12}^{j=12} y_{i+j}$ . This is easiest using `forvalues`. Plot this moving average against `hours` using the `twoway connected` graph command. Compare with the lowess plot.



# **Chapter 3**

## **Linear regression basics**

### 3.1 Introduction

Linear regression analysis is often the starting point of an empirical investigation. Because of its relative simplicity, it is useful for illustrating the different steps of a typical modeling cycle that involves an initial specification of the model followed by estimation, diagnostic checks, and model respecification. The purpose of such a linear regression analysis may be to summarize the data, generate conditional predictions, or test and evaluate the role of specific regressors. We will illustrate these aspects using a specific data example.

This chapter is limited to basic linear regression analysis on cross-sectional data of a continuous dependent variable. The setup is for a single equation and exogenous regressors. Some standard complications of linear regression, such as misspecification of the conditional mean and model errors that are heteroskedastic, will be considered. In particular, we model the natural logarithm of medical expenditures instead of the level. We will ignore other various aspects of the data that can lead to more sophisticated nonlinear models presented in later chapters.

## 3.2 Data and data summary

The first step is to decide what dataset will be used. In turn, this decision depends on the population of interest and the research question itself. We discussed how to convert a raw dataset to a form amenable to regression analysis in section [2.4](#). In this section, we present ways to summarize and gain some understanding of the data, a necessary step before any regression analysis.

### 3.2.1 Data description

We analyze medical expenditures in 2003 of individuals 65 years and older who qualify for healthcare under the U.S. Medicare program. The original data source is the Medical Expenditure Panel Survey.

Medicare does not cover all medical expenses. For example, copayments for medical services and expenses of prescribed pharmaceutical drugs were not covered for the time period studied here. About half of eligible individuals therefore purchase supplementary insurance in the private market that provides insurance coverage against various out-of-pocket expenses.

In this chapter, we consider the impact of this supplementary insurance on total annual medical expenditures of an individual, measured in dollars. A formal investigation must control for the influence of other factors that also determine individual medical expenditure, notably, sociodemographic factors such as age, gender, education and income, geographical location, and health-status measures such as self-assessed health and presence of chronic or limiting conditions. In this chapter, as in other chapters, we instead deliberately use a short list of regressors. This permits shorter output and simpler discussion of the results, an advantage because our intention is to simply explain the methods and tools available in Stata.

### 3.2.2 Variable description

Given the Stata dataset for analysis, we begin by using the `describe` command to list various features of the variables to be used in the linear

regression. The command without a variable list describes all the variables in the dataset. Here we restrict attention to the variables used in this chapter.

```
. * Variable description for medical expenditure dataset
. use mus203mepsmedexp
(A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e)
. describe totexp ltotexp posexp suppins phyylim actlim totchr age female income
```

Variable name	Storage type	Display format	Value label	Variable label
totexp	double	%12.0g		Total medical expenditure
ltotexp	float	%9.0g		$\ln(\text{totexp})$ if $\text{totexp} > 0$
posexp	float	%9.0g	posexp	Total expenditure > 0
suppins	float	%9.0g	suppins	Has supp priv insurance
phyylim	double	%12.0g	phyylim	Has functional limitation
actlim	double	%12.0g	actlim	Has activity limitation
totchr	double	%12.0g		# of chronic problems
age	double	%12.0g		Age
female	double	%12.0g	female	Female
income	double	%12.0g		Annual household income/1000

The variable types and format columns indicate that all the data are numeric. In this case, some variables are stored in single precision (`float`) and some in double precision (`double`). From the variable labels, we expect `totexp` to be nonnegative; `ltotexp` to be missing if `totexp` equals 0; `posexp`, `suppins`, `phyylim`, `actlim`, and `female` to be 0 or 1; `totchr` to be a nonnegative integer; `age` to be positive; and `income` to be nonnegative or positive. Note that the integer variables could have been stored much more compactly as `integer` or `byte`. The variable labels provide a short description that is helpful but may not fully describe the variable. For example, the key regressor `suppins` was created by aggregating across several types of private supplementary insurance.

### 3.2.3 Summary statistics

It is essential in any data analysis to first check the data by using the `summarize` command.

```
. * Summary statistics for medical expenditure dataset
. summarize totexp ltotexp posexp suppins phylim actlim totchr age female income
```

Variable	Obs	Mean	Std. dev.	Min	Max
totexp	3,064	7030.889	11852.75	0	125610
ltotexp	2,955	8.059866	1.367592	1.098612	11.74094
posexp	3,064	.9644256	.1852568	0	1
suppins	3,064	.5812663	.4934321	0	1
phylim	3,064	.4255875	.4945125	0	1
actlim	3,064	.2836162	.4508263	0	1
totchr	3,064	1.754243	1.307197	0	7
age	3,064	74.17167	6.372938	65	90
female	3,064	.5796345	.4936982	0	1
income	3,064	22.47472	22.53491	-1	312.46

On average, 96% of individuals incur medical expenditures during a year; 58% have supplementary insurance; 43% have functional limitations; 28% have activity limitations; and 58% are female because the elderly population is disproportionately female because of the greater longevity of women. The only variable to have missing data is `ltotexp`, the natural logarithm of `totexp`, which is missing for the  $(3064 - 2955) = 109$  observations with `totexp = 0`.

All variables have the expected range, except that income is negative. To see how many observations on `income` are negative, we use the `tabulate` command, restricting attention to nonpositive observations to limit output.

```
. * Tabulate variable
. tabulate income if income <= 0
```

Annual household income/1000	Freq.	Percent	Cum.
-1	1	1.14	1.14
0	87	98.86	100.00
Total	88	100.00	

Only one observation is negative, and negative income is possible for income from self-employment or investment. We include the observation in the analysis here, though checking the original data source may be warranted.

Much of the subsequent regression analysis will drop the 109 observations with 0 medical expenditures, so in a research article, it would be best to report summary statistics without these observations.

### 3.2.4 More detailed summary statistics

Additional descriptive analysis of key variables, especially the dependent variable, is useful. For `totexp`, the level of medical expenditures, `summarize, detail` yields

```
. * Detailed summary statistics of a single variable
. summarize totexp, detail
```

Total medical expenditure

	Percentiles	Smallest		
1%	0	0		
5%	112	0		
10%	393	0	Obs	3,064
25%	1271	0	Sum of wgt.	3,064
50%	3134.5		Mean	7030.889
		Largest	Std. dev.	11852.75
75%	7151	104823		
90%	17050	108256	Variance	1.40e+08
95%	27367	123611	Skewness	4.165058
99%	62346	125610	Kurtosis	26.26796

Medical expenditures vary greatly across individuals, with a standard deviation of 11,853, which is almost twice the mean. The median of 3,135 is much smaller than the mean of 7,031, reflecting the skewness of the data. For variable  $x$ , the skewness statistic is a scale-free measure of skewness that estimates  $E[\{(x - \mu)/\sigma\}^3] = E\{(x - \mu)^3\}/\sigma^3$ , the third central moment standardized by the cube of the standard deviation. The skewness is zero for symmetrically distributed data. The value here of 4.17 indicates considerable right skewness. The kurtosis statistic is an estimate of  $E[\{(x - \mu)/\sigma\}^4] = E\{(x - \mu)^4\}/\sigma^4$ , the fourth central moment standardized by the fourth power of the standard deviation. The reference value is 3, the value for normally distributed data. The much higher value here of 26.27 indicates that the tails are much thicker than those of a normal distribution. You can obtain additional summary statistics by using the

`centile` command to obtain other percentiles and by using the `table` command, which is explained in section [3.2.6](#).

We conclude that the distribution of the dependent variable is considerably skewed and has thick tails. These complications often arise for commonly studied individual-level economic variables such as expenditures, income, earnings, wages, and house prices. It is possible that including regressors will eliminate the skewness, but in practice, much of the variation in the data will be left unexplained ( $R^2 < 0.3$  is common for individual-level data), and skewness and excess kurtosis will remain.

Such skewed, thick-tailed data suggest a model with multiplicative errors instead of additive errors. A standard solution is to transform the dependent variable by taking the natural logarithm. Here this is complicated by the presence of 109 0-valued observations. We take the expedient approach of dropping the zero observations from analysis in either logs or levels. This should make little difference here because only 3.6% of the sample is then dropped. A better approach, using two-part or selection models, is covered in sections 19.5–19.7.

The output for `tabstat` in section [3.2.6](#) reveals that taking the natural logarithm for these data essentially eliminates the skewness and excess kurtosis.

The community-contributed `fsum` command ([Wolfe 2002](#)) is an enhancement of `summarize` that enables formatting the output and including additional information such as percentiles and variable labels. The community-contributed `outsum` command ([Papps 2006](#)) produces a text file of means and standard deviations for one or more subsets of the data, for example, one column for the full sample, one for a male subsample, and one for a female subsample.

### 3.2.5 Tables of frequencies

One-way tables can be created by using the `tabulate` command, presented in section [3.2.3](#), the `table` command, and the `tabstat` command. Two-way tables can also be created by using these commands.

For two-way tables of frequencies, only `table` produces clean output.  
For example,

```
. * Two-way table of frequencies  
. table female totchr
```

		# of chronic problems								
		0	1	2	3	4	5	6	7	Total
Female										
No		239	415	323	201	82	23	4	1	1,288
Yes		313	466	493	305	140	46	11	2	1,776
Total		552	881	816	506	222	69	15	3	3,064

provides frequencies for a two-way tabulation of gender against the number of chronic conditions. The option `stat(percent)` provides percentages rather than frequencies.

The `tabulate` command can provide both row and column percentages.  
For example,

```
. * Two-way table with row and column percentages and Pearson chi-squared
. tabulate female suppins, row col chi2
```

Key
<i>frequency</i>
<i>row percentage</i>
<i>column percentage</i>

Female	Has supp priv insurance		Total
	No	Yes	
No	488	800	1,288
	37.89	62.11	100.00
	38.04	44.92	42.04
Yes	795	981	1,776
	44.76	55.24	100.00
	61.96	55.08	57.96
Total	1,283	1,781	3,064
	41.87	58.13	100.00
	100.00	100.00	100.00

Pearson chi2(1) = 14.4991 Pr = 0.000

Comparing the row percentages for this sample, we see that while a woman is more likely to have supplemental insurance than not, the probability that a woman in this sample has purchased supplemental insurance is lower than the probability that a man in this sample has purchased supplemental insurance. Although we do not have the information to draw these inferences for the population, the results for Pearson's chi-squared test soundly reject the null hypothesis that these variables are independent. Other tests of association are available. The related command `tab2` will produce all possible two-way tables that can be obtained from a list of several variables.

For multiway tables, it is best to use `table`. For the example at hand, we have

```
. * Three-way table of frequencies
. table female suppins totchr, nototals
```

		# of chronic problems							
		0	1	2	3	4	5	6	7
Female	No								
	Has supp priv insurance								
	No	102	165	121	68	25	6	1	
	Yes	137	250	202	133	57	17	3	1
Yes									
	Has supp priv insurance								
	No	135	212	233	134	56	22	1	2
	Yes	178	254	260	171	84	24	10	

An alternative is to use `tabulate` with the `by` prefix, but the results are not as neat as those from `table`.

### 3.2.6 Tables of summary statistics

The preceding tabulations will produce voluminous output if one of the variables being tabulated takes on many values. Then it is much better to use command `table` with the `statistics()` option to present tables that give key summary statistics for that variable, such as the mean and standard deviation. Note that the `statistics()` option, abbreviated `stat()`, was introduced in Stata 17 and replaces the `contents()` option available in earlier versions of Stata. Such tabulations can be useful even when variables take on few values. For example, when summarizing the number of chronic problems by gender, `table` yields

```
. * One-way table of summary statistics
. table (result) female, stat(count totchr) stat(mean totchr) stat(sd totchr)
>     stat(p50 totchr)
```

	Female		
	No	Yes	Total
Number of nonmissing values	1,288	1,776	3,064
Mean	1.659938	1.822635	1.754243
Standard deviation	1.261175	1.335776	1.307197
50th percentile	1	2	2

Women on average have more chronic problems (1.82 versus 1.66 for men). The option `stat()` can produce many other statistics, including the minimum, maximum, and key percentiles.

The `table` command with the `stat()` options can additionally produce two-way and multiway tables of summary statistics. As an example,

```
. * Two-way table of summary statistics
. table female suppins, stat(count totchr) stat(mean totchr) nototals
```

		Has supp insurance	
		No	Yes
Female			
No		488	800
Number of nonmissing values		1.530738	1.73875
Mean			
Yes		795	981
Number of nonmissing values		1.803774	1.83792
Mean			

shows that those with supplementary insurance on average have more chronic problems. This is especially so for males (1.74 versus 1.53).

The `tabulate`, `summarize()` command can be used to produce one-way and two-way tables with means, standard deviations, and frequencies. This is a small subset of the statistics that can be produced using `table`, so we might as well use `table`.

The `tabstat` command provides a table of summary statistics that permits more flexibility than `summarize`. The following output presents summary statistics on medical expenditures and the natural logarithm of expenditures that are useful in determining skewness and kurtosis.

```
. * Summary statistics obtained using command tabstat
. tabstat totexp ltotexp, statistics(count mean p50 sd skew kurt)
> columns(statistics)
```

Variable	N	Mean	p50	SD	Skewness	Kurtosis
totexp	3064	7030.889	3134.5	11852.75	4.165058	26.26796
ltotexp	2955	8.059866	8.111928	1.367592	-.3857887	3.842263

This reproduces information given in section [3.2.4](#) and shows that taking the natural logarithm eliminates most skewness and kurtosis. The `columns(statistics)` option presents the results with summary statistics being given in the columns and each variable being given in a separate row. Without this option, we would have summary statistics in rows and variables in the columns. A two-way table of summary statistics can be obtained by using the `by()` option.

The `collect` command, introduced in Stata 17, provides great flexibility in creating production-quality tables. The command is illustrated in section [3.5.7](#).

### 3.2.7 Hypothesis tests on the population mean

The `ttest` command can be used to test hypotheses about the population mean of a single variable ( $H_0: \mu = \mu^*$  for specified value  $\mu^*$ ) and to test the equality of means ( $H_0: \mu_1 = \mu_2$ ). For more general analysis of variance and analysis of covariance, the `oneway` and `anova` commands can be used, and several other tests exist for more specialized examples such as testing the equality of proportions.

These commands are rarely used in microeconomics because they can be recast as a special case of regression with an intercept and appropriate indicator variables. Furthermore, regression has the advantage of reliance on less restrictive distributional assumptions, provided samples are large enough for asymptotic theory to provide a good approximation.

For examples of the `ttest` command and comparison with tests based on OLS estimation, see section [3.5.12](#).

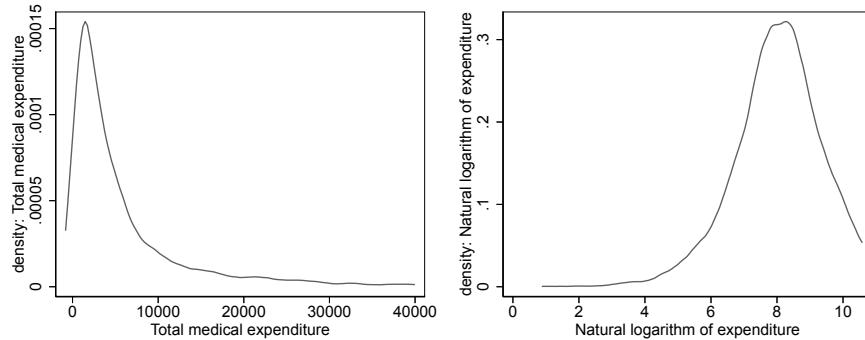
### 3.2.8 Data plots

It is useful to plot a smoothed histogram or a density estimate of the dependent variable. Here we use the `kdensity` command, which provides a kernel estimate of the density.

The data are highly skewed, with a 97th percentile of approximately \$40,000 and a maximum of \$125,000. The `kdensity totexp` command will therefore bunch 97% of the density in the first 30% of the  $x$  axis. One possibility is to type `kdensity totexp if totexp < 40000`, but this produces a kernel density estimate assuming the data are truncated at \$40,000. Instead, we use command `kdensity totexp`, we save the evaluation points in `kx1` and the kernel density estimates in `kd1`, and then we line-plot `kd1` against `kx1`.

We do this for both the level and the natural logarithm of medical expenditures, and we use `graph combine` to produce a figure that includes both density graphs (shown in figure 3.1). We have

```
. * Kernel density plots with adjustment for highly skewed data
. kdensity totexp if posexp==1, generate(kx1 kd1) n(500)
. graph twoway (line kd1 kx1) if kx1 < 40000, name(levels, replace)
. label variable ltotexp "Natural logarithm of expenditure"
. kdensity ltotexp if posexp==1, generate(kx2 kd2) n(500)
. graph twoway (line kd2 kx2) if kx2 < ln(40000), name(logs, replace)
. graph combine levels logs, iscale(1.2) ysize(2.5) xsize(6.0)
```



**Figure 3.1.** Comparison of densities of level and natural logarithm of medical expenditures

Only positive expenditures are considered, and for graph readability, the very long right tail of `totexp` has been truncated at \$40,000. In figure 3.1, the distribution of `totexp` is very right skewed, whereas that of `ltotexp` is fairly symmetric.

### 3.3 Transformation of data before regression

When one specifies a linear regression model, the presumption is that the specified relationship between the variable of interest  $y$  and the regressors  $\mathbf{x}$  is linear, which means that the marginal response of  $y$  to a unit change in  $x$  is constant.

The preferred model linking  $y$  and the regressors, however, may not be linear. For example, the relation between total production costs and output is usually specified to be nonlinear. In such cases, it is usual to interpret the regression as linear after transformation from the original units.

Transformations to the linear form may involve both  $y$  and  $\mathbf{x}$ , or just one of those components.

The purpose of the transformation is to “straighten out” a relationship. Consider some leading examples. Suppose that the relationship takes the form  $y = \exp(\beta_1 + \beta_2 x + u)$ , where  $x$  denotes the regressor and  $u$  is the error term. Then the transformation  $\ln y = \beta_1 + \beta_2 x + u$  is a “semilog” or “log-linear” regression that relates  $\ln y$  to  $x$ . After transformation,  $\beta_2$  measures  $\partial E(\ln y)/\partial x = (1/y) \times \partial y/\partial x$ , which varies inversely with  $y$ .

Now consider the multiplicative relationship  $y = e^{\beta_1} x^{\beta_2} u$ . Taking logs on both sides of the equality yields  $\ln y = \beta_1 + \beta_2 \ln x + \ln u$ , a linear-in-logs or log–log regression. In this case, the coefficient  $\beta_1$  measures  $\partial E\{\ln(y)/\partial \ln x\}$ , that is, the elasticity of  $y$  with respect to  $x$  based on the constant elasticity model.

In both the preceding examples, while the dependent variable and, in the second example, the regressor have been transformed, the transformed models are linear in the parameters. So the transformed models can be fit using OLS regression, the subject of this chapter.

More generally, we may consider a regression such as  $g(y) = f(x, \beta) + u$ , where  $g(\cdot)$  and  $f(\cdot)$  denote some linearizing transformation whose specific form will depend upon the context. Choosing the functions that provide the best approximations to the data-generating

process (DGP) is a part of model specification. Having chosen one, one relies on statistical tests to check whether the functional form is such that the remaining unexplained variation is roughly random.

Although the least-squares estimator of the linear regression requires only the data, or the error on the regression, to have quite weak distributional properties, transformations are often motivated by a preference for some particular features. For example, some outcomes such as income and expenditure often display a highly skewed distribution. A log transformation will typically make the distribution more symmetric and less nonnormal.

Another motivation for transformation is to make the error variance less heteroskedastic. For example, in its original form, a regression may display dependence between (say) the location parameter  $E(y|x)$  and scale parameter  $\text{Var}(y|x)$ ; a transformation may get rid of such dependence by reducing the heteroskedasticity of the error term. A family of power transformations, known as Box–Cox transformations, that replaces  $y$  by  $y^p$  is motivated by a similar consideration. A special case is  $p = 1/2$ , the square-root transformation. In a third example, suppose  $y$  is positive and we want to ensure that fitted values of  $y$  remain positive. A log transformation ensures this. In the final example, suppose  $y$  is a proportion, that is,  $0 < y < 1$ , and again we want the fitted values from the regression to preserve this property, whereas the linear regression of  $y$  on  $x$  will not. The logit transformation uses the transformed dependent variable  $\log\{y/(1 - y)\}$ , which satisfies this requirement. This transformation also changes the range of values of the dependent variable, producing greater symmetry and spread in the tails of the distribution. In some cases, such changes make the least-squares estimator more robust.

Transformations generally affect the interpretations of regression coefficients, and transformations involving the dependent variables will also affect measures of goodness of fit such as regression  $R^2$ . This means that regression statistics such as  $R^2$  with  $g(y)$  as a dependent variable cannot be directly compared with those with  $y$  as the dependent variable. This complicates the comparison of regressions with different transformations of the dependent variable. A substantial literature exists on

the topic of comparison of linear and linear-in-logs regressions; see [Godfrey and Wickens \(1981\)](#) and references cited there.

Finally, even if one chooses to regress  $g(y)$  on  $h(x)$ , one may want to interpret the results in terms of the original units of  $y$  and  $x$ . This involves a thorny problem of *retransformation* that is discussed in section [4.2.3](#). In some cases, retransformation can be avoided by directly modeling  $y$  using methods more advanced than OLS regression. In particular, we can use Poisson regression (`poisson` command) in place of the log-linear model and use logit regression (`logit` command) for proportions data.

Economic theory rarely suggests a specific parametric form of a regression model, thereby leaving room for empirical explorations. Nonparametric regressions (see section [14.6](#)) are less restrictive in this respect.

## 3.4 Linear regression

We present the linear regression model, first in levels and then for a transformed dependent variable, here in logs.

### 3.4.1 Basic regression theory

We begin by introducing terminology used throughout the rest of this book. Let  $\theta$  denote the vector of parameters to be estimated, and let  $\hat{\theta}$  denote an estimator of  $\theta$ . Ideally, the distribution of  $\hat{\theta}$  is centered on  $\theta$  with small variance, for precision, and a known distribution, to permit statistical inference. We restrict analysis to estimators that are consistent for  $\theta$ , meaning that in infinitely large samples,  $\hat{\theta}$  equals  $\theta$  aside from negligible random variation. This is denoted by  $\hat{\theta} \xrightarrow{p} \theta$  or, more formally, by  $\hat{\theta} \xrightarrow{p} \theta_0$ , where  $\theta_0$  denotes the unknown “true” parameter value. A necessary condition for consistency is correct model specification or, in some leading cases, correct specification of key components of the model, most notably the conditional mean.

Under additional assumptions, most of the estimators considered in this book are asymptotically normally distributed, meaning that their distribution is well approximated by the multivariate normal in large samples. This is denoted by

$$\hat{\theta} \xrightarrow{a} N \left\{ \theta, \text{Var}(\hat{\theta}) \right\}$$

where  $\text{Var}(\hat{\theta})$  denotes the (asymptotic) variance–covariance matrix of the estimator (VCE). More efficient estimators have smaller VCES. The VCE depends on unknown parameters, so we use an estimate of the VCE, denoted by  $\hat{V}(\hat{\theta})$ . Standard errors of the parameter estimates are obtained as the square root of diagonal entries in  $\hat{V}(\hat{\theta})$ . Different assumptions about the DGP, such as heteroskedasticity, can lead to different estimates of the VCE.

Test statistics based on asymptotic normal results lead to the use of the standard normal distribution and chi-squared distribution to compute critical values and  $p$ -values. For some estimators, notably, the OLS estimator, tests are instead based on the  $t$  distribution and the  $F$  distribution. This makes essentially no difference in large samples with, say, degrees of freedom greater than 100, but in practice it provides a better approximation especially for cluster-robust inference with few clusters; see section [3.4.6](#).

### 3.4.2 OLS regression and matrix algebra

The goal of linear regression is to estimate the parameters of the linear conditional mean

$$E(y|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta} = \beta_1x_1 + \beta_2x_2 + \cdots + \beta_Kx_K \quad (3.1)$$

where usually an intercept is included so that  $x_1 = 1$ . Here  $\mathbf{x}$  is a  $K \times 1$  column vector with the  $j$ th entry—the  $j$ th regressor  $x_j$ —and  $\boldsymbol{\beta}$  is a  $K \times 1$  column vector with the  $j$ th entry  $\beta_j$ .

Sometimes,  $E(y|\mathbf{x})$  is of direct interest for prediction. More often, however, econometrics studies are interested in one or more of the associated marginal effects (MES),

$$\frac{\partial E(y|\mathbf{x})}{\partial x_j} = \beta_j$$

for the  $j$ th regressor. For example, we are interested in the MES of supplementary private health insurance on medical expenditures. An attraction of the linear model is that estimated MES are given directly by estimates of the slope coefficients.

The linear regression model specifies an additive (often specified to be independent and identically distributed) error so that, for the typical  $i$ th observation,

$$y_i = \mathbf{x}'_i \boldsymbol{\beta} + x u_i, \quad i = 1, \dots, N$$

The OLS estimator minimizes the sum of squared errors,  $\sum_{i=1}^N (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2$ .

Matrix notation provides a compact way to represent the estimator and variance matrix formulas that involve sums of products and cross products. We define the  $N \times 1$  column vector  $\mathbf{y}$  to have the  $i$ th entry  $y_i$ , and we define the  $N \times K$  regressor matrix  $\mathbf{X}$  to have the  $i$ th row  $\mathbf{x}'_i$ . Then the OLS estimator can be written in several ways, with

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \\ &= \left( \sum_{i=1}^N \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \sum_{i=1}^N \mathbf{x}_i y_i \\ &= \left[ \begin{array}{cccc} \sum_{i=1}^N x_{1i}^2 & \sum_{i=1}^N x_{1i}x_{2i} & \cdots & \sum_{i=1}^N x_{1i}x_{Ki} \\ \sum_{i=1}^N x_{2i}x_{1i} & \sum_{i=1}^N x_{2i}^2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \sum_{i=1}^N x_{Ki}x_{1i} & \cdots & \cdots & \sum_{i=1}^N x_{Ki}^2 \end{array} \right]^{-1} \left[ \begin{array}{c} \sum_{i=1}^N x_{1i}y_i \\ \sum_{i=1}^N x_{2i}y_i \\ \vdots \\ \sum_{i=1}^N x_{Ki}y_i \end{array} \right]\end{aligned}$$

We define all vectors as column vectors, with a transpose if row vectors are desired. By contrast, Stata commands and Mata commands define vectors as row vectors, so in parts of Stata and Mata code, we need to take a transpose to conform to the notation in the book.

### 3.4.3 Properties of the OLS estimator

The properties of any estimator vary with the assumptions made about the DGP. For the linear regression model, this reduces to assumptions about the regression error  $u_i$ .

The starting point for analysis is to assume that  $u_i$  satisfies the following classical conditions:

1.  $E(u_i | \mathbf{x}_i) = 0$  (exogeneity of regressors)
2.  $E(u_i^2 | \mathbf{x}_i) = \sigma^2$  (conditional homoskedasticity)
3.  $E(u_i u_j | \mathbf{x}_i, \mathbf{x}_j) = 0, i \neq j$  (conditionally uncorrelated observations)

Assumption 1 is essential for consistent estimation of  $\beta$  and implies that the conditional mean given in (3.1) is correctly specified. This means that the conditional mean is linear and that all relevant variables have been included in the regression. Assumption 1 is relaxed in chapter 7. Assumptions 2 and 3 determine the form of the VCE of  $\hat{\beta}$ .

### 3.4.4 Default standard errors

Assumptions 1–3 lead to  $\hat{\beta}$  being asymptotically normally distributed with the default estimator of the VCE

$$\hat{V}_{\text{default}}(\hat{\beta}) = s^2(\mathbf{X}'\mathbf{X})^{-1}$$

where

$$s^2 = (N - K)^{-1} \sum_{i=1}^N \hat{u}_i^2 \tag{3.2}$$

and  $\hat{u}_i = y_i - \mathbf{x}'_i \hat{\beta}$ . Under assumptions 1–3, the OLS estimator is fully efficient. If, additionally,  $u_i$  is normally distributed, then “ $t$  statistics” are exactly  $t$  distributed. This fourth assumption is not made, but it is common to continue to use the  $t$  distribution in the hope that it provides a better approximation than the standard normal in finite samples.

When assumptions 2 and 3 are relaxed, OLS is no longer fully efficient. In chapter 6, we present examples of more efficient, feasible generalized least-squares estimation. In the current chapter, we continue to use the OLS estimator, as is often done in practice, but we use alternative estimates of the VCE that are valid when assumption 2, assumption 3, or both are relaxed, provided the sample size is sufficiently large for the relevant asymptotic theory to provide a good approximation.

### 3.4.5 Heteroskedasticity-robust standard errors

Given assumptions 1 and 3, but not 2, we have heteroskedastic uncorrelated errors. Then a robust estimator, or more precisely a heteroskedasticity-robust estimator, of the VCE of the OLS estimator is

$$\widehat{V}_{\text{robust}}(\widehat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1} \left( \frac{N}{N-K} \sum_{i=1}^N \widehat{u}_i^2 \mathbf{x}_i \mathbf{x}_i' \right) (\mathbf{X}'\mathbf{X})^{-1} \quad (3.3)$$

For cross-sectional data that are independent, this estimator, introduced by [White \(1980\)](#), has supplanted the default variance matrix estimate in most applied work because heteroskedasticity is the norm, and in that case, the default estimate of the VCE is incorrect.

In Stata, a robust estimate of the VCE is obtained by using the `vce(robust)` option of the `regress` command, as illustrated in section [3.5.2](#). Related options are `vce(hc2)` and `vce(hc3)`, which may provide better heteroskedasticity-robust estimates of the VCE when the sample size is small; see [R] `regress`. The robust estimator of the VCE has been extended to other estimators and models, and a feature of Stata is the `vce(robust)` option, which is applicable for many estimation commands. Some community-contributed commands use `robust` in place of `vce(robust)`.

### 3.4.6 Cluster-robust standard errors

When errors for different observations are correlated, assumption 3 is violated. Then both default and heteroskedastic robust estimates of the VCE are invalid, and different ways in which error correlation may arise lead to different robust estimates of the VCE. Various robust estimates of the VCE are presented in section [13.4](#).

For cross-sectional data, the most common violation of assumption 3 is that errors are clustered. Clustered or grouped errors are errors that are correlated within a cluster or group and are uncorrelated across clusters. A simple example of clustering arises when sampling is of independent units but errors for individuals within the unit are correlated. For example, 100 independent villages may be sampled, with several people from each village surveyed. Then, if a regression model overpredicts  $y$  for one village member, it is likely to overpredict for other members of the same village, indicating positive correlation. Similar comments apply when sampling is of households with several individuals in each household. Another leading example is panel data with independence over individuals but with correlation over time for a given individual.

Given assumption 1, but not 2 or 3, a cluster-robust estimator of the VCE of the OLS estimator is

$$\widehat{V}_{\text{cluster}} (\widehat{\beta}) = (\mathbf{X}' \mathbf{X})^{-1} \left( \frac{G}{G-1} \frac{N-1}{N-K} \sum_{g=1}^G \mathbf{X}_g \widehat{\mathbf{u}}_g \widehat{\mathbf{u}}_g' \mathbf{X}_g' \right) (\mathbf{X}' \mathbf{X})^{-1}$$

where  $g = 1, \dots, G$  denotes the cluster (such as village),  $\widehat{\mathbf{u}}_g$  is the vector of residuals for the observations in the  $g$ th cluster, and  $\mathbf{X}_g$  is a matrix of the regressors for the observations in the  $g$ th cluster. The key assumptions made are error independence across clusters and that the number of clusters  $G \rightarrow \infty$ .

Cluster-robust standard errors can be computed by using the `vce(cluster clustvar)` option in Stata, where clusters are defined by the different values taken by the *clustvar* variable. The estimate of the VCE is in fact heteroskedasticity-robust and cluster-robust because there is no

restriction on  $\text{Cov}(u_{gi}, u_{gj})$ . The cluster vce estimate can be applied to many estimators and models; see section [13.4.6](#).

Cluster-robust standard errors must be used when data are clustered. For a scalar regressor  $x$ , a rule of thumb is that cluster-robust standard errors are

$$\tau \simeq \sqrt{1 + \rho_x \rho_u (M - 1)} \quad (3.4)$$

times the incorrect default standard errors, where  $\rho_x$  is the within-cluster correlation coefficient of the regressor,  $\rho_u$  is the within-cluster correlation coefficient of the error, and  $M$  is the average cluster size. This rule of thumb is a good guide in most settings, but when  $x$  is an experimentally assigned treatment with values that vary across observations within the same cluster, one should use the more general rule of thumb that  $\tau \simeq \sqrt{1 + \rho_{xu} (M - 1)}$ , where  $\rho_{xu}$  is the within-cluster correlation of  $x_i u_i$ . Cluster-robust standard errors can be much larger than default or heteroskedastic-robust standard errors.

It can be necessary to use cluster-robust standard errors even where it is not immediately obvious. This is particularly the case when a regressor is an aggregated or macrovariable because then  $\rho_x = 1$ . For example, suppose we use data from the U.S. Current Population Survey and regress individual earnings on individual characteristics and a state-level regressor that does not vary within a state. Then, if there are many individuals in each state so  $M$  is large, even slight error correlation for individuals in the same state can lead to great downward bias in default standard errors and in heteroskedasticity-robust standard errors. Clustering can also be induced by the design of sample surveys. This topic is pursued in section [6.9](#).

Statistical inference for OLS based on cluster-robust standard errors uses critical values and  $p$ -values based on the  $t$  distribution with  $(G - 1)$  degrees of freedom, where  $G$  is the number of clusters. When there are few clusters, this approximation can lead to considerable underestimation of standard errors and associated test  $p$ -values and to confidence intervals that are too narrow. Better inference for OLS with few clusters is pursued in

section [6.4.6](#) and in section [12.6](#). In particular, see section [12.6](#) for the community-contributed `boottest` command ([Roodman et al. 2019](#)), which implements a wild cluster bootstrap that can lead to better finite cluster inference.

Many microeconometric applications use clustered data. Then other estimators than OLS are often used, most notably fixed-effects and random-effects estimators. For linear models, these methods are presented in sections [6.5–6.7](#) and, for panel data, in chapter [8](#). For nonlinear models, see section [13.9](#) and, for panel data, see chapter 22. For the recently proposed design-based approach to inference, see section 24.4.7.

### 3.4.7 Bootstrap standard errors

An appropriate alternative way to compute heteroskedasticity-robust or cluster-robust standard errors is to use an appropriate bootstrap. This is a widely applicable method for obtaining standard errors and confidence intervals for parameters in cases where the asymptotic distribution is either not available or is available but is inconvenient to implement.

Here we present simple bootstraps that yield standard errors that are asymptotically equivalent to those obtained using the `vce(robust)` and `vce(cluster clustvar)` options. A refined bootstrap procedure, if feasible, provides an improvement over the usual asymptotic distribution. These distinctions are further developed and used in section [12.5](#).

The basic idea of the bootstrap is that the sample is used as a population, and we then obtain a number of samples from this “population” by repeatedly resampling observations with replacement. Such samples are referred to as bootstrap samples. This is a substitute for the ideal but impractical situation of having multiple independent samples. We then obtain the sampling distribution of the parameters of interest by fitting the same model to the many bootstrap samples. Moments of the distribution can then be computed from the collection of estimates.

Resampling from a given sample is easiest to understand in the independent and identically distributed setting with sample  $y_i, i = 1, \dots, N$

. Suppose that the target parameter is the population mean, denoted  $\mu$ , and the estimator  $\hat{\mu}$  is the sample mean  $\bar{y}$ . Then we can draw  $B$  different samples of  $N$  observations each by sampling with replacement. Each sample generates a sample mean,  $\bar{y}_b$ ,  $b = 1, \dots, B$ , so we have  $B$  independent estimates. Moments of the distribution of  $\hat{\mu}$  can then be computed given the empirical distribution of these  $B$  estimates.

Now consider the linear regression setting with data  $(y_i, \mathbf{x}_i)$ ,  $i = 1, \dots, N$ , and model errors that are independent but heteroskedastic. A bootstrap called a paired bootstrap or nonparametric bootstrap obtains bootstrap resamples by sampling  $(y_i, \mathbf{x}_i)$ , jointly and with replacement. Each bootstrap sample of  $N$  observations generates an estimate of the regression parameters, denoted  $\hat{\beta}_b$ ,  $b = 1, \dots, B$ .

The bootstrap estimate of variance of an estimator is the usual formula for estimating a variance of (say)  $\beta_j$ , applied to the  $B$  bootstrap replications

$$s_{\hat{\beta}_j}^2 = \frac{1}{B-1} \sum_b \left( \hat{\beta}_{jb} - \bar{\hat{\beta}}_{jb} \right)^2$$

The bootstrap  $100(1 - \alpha)$  percent confidence interval for  $\beta_j$  is obtained by using the asymptotic  $\alpha$  percent critical values from the standard normal distribution,

$$\left( \hat{\beta}_j - z_{\alpha/2} \times s_{\hat{\beta}_j}, \hat{\beta}_j + z_{\alpha/2} \times s_{\hat{\beta}_j} \right)$$

where  $\Pr(Z > z_{\alpha/2}) = \alpha/2$ . This bootstrap yields standard errors and confidence intervals that are asymptotically equivalent to those obtained using heteroskedastic–robust standard errors.

When regression model errors are instead clustered, the preceding method is adapted by resampling entire clusters with replacement. Each bootstrap sample of  $G$  clusters generates an estimate of the regression

parameters, denoted  $\widehat{\beta}_b$ ,  $b = 1, \dots, B$ . Then  $s_{\widehat{\beta}_j}^2$  and the associated confidence interval are computed using the preceding formulas. This cluster pairs bootstrap yields standard errors and confidence intervals that are equivalent as  $G \rightarrow \infty$  to those obtained using cluster-robust standard errors.

### 3.4.8 Regression in logs

The medical expenditure data are very right skewed. Then a linear model in levels can provide very poor predictions because it restricts the effects of regressors to be additive. For example, aging 10 years is assumed to increase medical expenditures by the same amount regardless of observed health status. Instead, it is more reasonable to assume that aging 10 years has a multiplicative effect. For example, it may increase medical expenditures by 20%.

We begin with an exponential mean model for positive expenditures, with error that is also multiplicative, so  $y_i = \exp(\mathbf{x}'_i \boldsymbol{\beta}) \varepsilon_i$ . Defining  $\varepsilon_i = \exp(u_i)$ , we have  $y_i = \exp(\mathbf{x}'_i \boldsymbol{\beta} + u_i)$ , and taking the natural logarithm, we fit the log-linear model

$$\ln y_i = \mathbf{x}'_i \boldsymbol{\beta} + u_i$$

by OLS regression of  $\ln y$  on  $\mathbf{x}$ . The conditional mean of  $\ln y$  is being modeled, rather than the conditional mean of  $y$ . In particular,

$$E(\ln y | \mathbf{x}) = \mathbf{x}' \boldsymbol{\beta}$$

assuming  $u_i$  has conditional mean zero.

Parameter interpretation requires care. For regression of  $\ln y$  on  $\mathbf{x}$ , the coefficient  $\beta_j$  measures the effect of a change in regressor  $x_j$  on  $E(\ln y | \mathbf{x})$ , but ultimate interest lies instead on the effect on  $E(y | \mathbf{x})$ . Some algebra

shows that  $\beta_j$  measures the proportionate change in  $E(y|\mathbf{x})$  as  $x_j$  changes, called a semielasticity, rather than the level of change in  $E(y|\mathbf{x})$ . For example, if  $\beta_j = 0.02$ , then a one-unit change in  $x_j$  is associated with a proportionate increase of 0.02, or a 2% increase, in  $E(y|\mathbf{x})$ .

Prediction of  $E(y|\mathbf{x})$  is substantially more difficult because it can be shown that  $E(\ln y|\mathbf{x}) \neq \exp(\mathbf{x}'\boldsymbol{\beta})$ . This is pursued in section [4.2.3. Buntin and Zaslavsky \(2004\)](#) compare several alternative regression models for medical expenditures.

## 3.5 Basic regression analysis

We use `regress` to run an OLS regression of the natural logarithm of medical expenditures, `ltotexp`, on `suppins` and several demographic and health-status measures. Using  $\ln y$  rather than  $y$  as the dependent variable leads to no change in the implementation of OLS but, as already noted, will change the interpretation of coefficients and predictions.

Many of the details we provide in this section are applicable to all Stata estimation commands, not just to `regress`.

### 3.5.1 Correlations

Before regression, it can be useful to investigate pairwise correlations of the dependent variables and key regressor variables by using `correlate`. We have

```
. * Pairwise correlations for dependent variable and regressor variables
. correlate ltotexp suppins phyylim actlim totchr age female income
(obs=2,955)
```

	ltotexp	suppins	phyylim	actlim	totchr	age	female
ltotexp	1.0000						
suppins	0.0941	1.0000					
phyylim	0.2924	-0.0243	1.0000				
actlim	0.2888	-0.0675	0.5904	1.0000			
totchr	0.4283	0.0124	0.3334	0.3260	1.0000		
age	0.0858	-0.1226	0.2538	0.2394	0.0904	1.0000	
female	-0.0058	-0.0796	0.0943	0.0499	0.0557	0.0774	1.0000
income	0.0023	0.1943	-0.1142	-0.1483	-0.0816	-0.1542	-0.1312
		income					
income			1.0000				

Medical expenditures are highly correlated with the health-status measures `phyylim`, `actlim`, and `totchr`. The regressors are only weakly correlated with each other, aside from the health-status measures. Note that `correlate` restricts analysis to the 2,955 observations where data are available for all variables in the variable list. The related command `pwcorr`, not

demonstrated, with the `sig` option gives the statistical significance of the correlations.

### 3.5.2 The `regress` command

The `regress` command performs OLS regression and yields an analysis-of-variance table, goodness-of-fit statistics, coefficient estimates, standard errors,  $t$  statistics,  $p$ -values, and confidence intervals. The syntax of the command is

```
regress depvar [indepvars] [if] [in] [weight] [, options]
```

Other Stata estimation commands have similar syntaxes. The output from `regress` is similar to that from many linear regression packages.

For independent cross-sectional data, the standard approach is to use the option `vce(robust)`, which gives standard errors that are valid even if model errors are heteroskedastic; see section [3.4.5](#).

When the `vce(robust)` option is used, the output from `regress` no longer includes the analysis-of-variance table presented in output for the section [1.6.2](#) example. The reason is that the  $F$  statistic for overall significance is calculated from sums of squares and degrees of freedom only in the special assumption of homoskedastic errors.

We obtain

```

. * OLS regression with heteroskedasticity-robust standard errors
. regress ltotexp suppins phyylim actlim totchr age female income, vce(robust)

Linear regression
Number of obs      =      2,955
F(7, 2947)        =     126.97
Prob > F          =     0.0000
R-squared          =     0.2289
Root MSE           =     1.2023

```

ltotexp	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
suppins	.2556428	.0465982	5.49	0.000	.1642744 .3470112
phyylim	.3020598	.057705	5.23	0.000	.1889136 .415206
actlim	.3560054	.0634066	5.61	0.000	.2316797 .4803311
totchr	.3758201	.0187185	20.08	0.000	.3391175 .4125228
age	.0038016	.0037028	1.03	0.305	-.0034587 .011062
female	-.0843275	.045654	-1.85	0.065	-.1738444 .0051894
income	.0025498	.0010468	2.44	0.015	.0004973 .0046023
_cons	6.703737	.2825751	23.72	0.000	6.149673 7.257802

The regressors are jointly statistically significant because the overall  $F$  statistic of 126.97 has a  $p$ -value of 0.000. At the same time, much of the variation is unexplained with  $R^2 = 0.2289$ . The root MSE statistic reports  $s$ , the standard error of the regression, defined in (3.2). For a two-sided test at level 0.05, a regressor is individually statistically significant if  $p < 0.05$ . Thus `age` and `female` are statistically insignificant, while the other variables are statistically significant at level 0.05. The statistical insignificance of age may be due to sample restriction to elderly people and the inclusion of several health-status measures that capture well the health effect of age.

Statistical significance of coefficients is easily established. More important is the economic significance of coefficients, meaning the measured impact of regressors on medical expenditures. This is straightforward for regression in levels because we can directly use the estimated coefficients. But here the regression is in logs. From section 3.4.8, in the log-linear model, parameters need to be interpreted as semielasticities. For example, the coefficient on `suppins` is 0.256. This means that private supplementary insurance is associated with a 0.256 proportionate rise, or a 25.6% rise, in medical expenditures. Similarly, large effects are obtained for the health-status measures, whereas health expenditures for women are 8.4% lower than those for men after controlling for other characteristics. The `income` coefficient of 0.0025 suggests a very small effect, but this is

misleading. The standard deviation of `income` is 23 (see section [3.2.3](#)), so a 1-standard deviation in `income` leads to a 0.058 proportionate rise, or 5.8% rise, in medical expenditures.

MES in nonlinear models are discussed in more detail in section [13.7](#). The preceding interpretations are based on calculus methods that consider very small changes in the regressor. For larger changes in the regressor, the finite-difference method is more appropriate. Then the interpretation in the log-linear model is similar to that for the exponential conditional mean model; see section [13.7.3](#). For example, the estimated effect of going from no supplementary insurance (`suppins=0`) to having supplementary insurance (`suppins=1`) is more precisely a  $100 \times (e^{0.256} - 1)$ , or 29.2%, rise.

The `ereturn list` command provides a list of what estimation results are stored in `e()`; see section [1.6.2](#) for details following the `regress` command. Stored results include regression coefficients in `e(b)` and the estimated VCE in `e(V)`.

Various postestimation commands, summarized next, enable prediction, computation of residuals, hypothesis testing, and model specification tests.

To see what stored results and postestimation commands are available, give commands

- . \* Display stored results and list available postestimation commands
- . `ereturn list`  
*(output omitted)*
- . `help regress postestimation`  
*(output omitted)*

### 3.5.3 Postestimation commands

Standard postestimation commands available after most estimation commands are given in table [3.1](#).

**Table 3.1.** Postestimation commands

Topic	Command
Results	<code>estat summarize, estat vce, estat ic, etable</code>
Store results	<code>estimates</code>
Prediction	<code>predict, predictnl</code>
MES	<code>margins, marginsplot</code>
Confidence intervals	<code>lincom, nlcom</code>
Hypothesis tests	<code>test, testnl, lrtest</code>
Specification tests	<code>hausman, linktest</code>

These postestimation commands are not only available following standard model commands such as `logit` and `poisson` but also available for the `n1` and `mlexp` commands for user-defined models. The `estat` commands provide extended statistics.

Other standard postestimation commands available following the `regress` command, and some other commands, are `contrast` for hypothesis tests, `pwcompare` for pairwise comparison of estimates, `suest` for seemingly unrelated regressions, and `forecast` for time-series regressions.

Postestimation commands specific to the `regress` command are `estat imtest` and `estat ovtest` for specification tests; `estat hettest` and `estat szroeter` for heteroskedasticity tests; `estat moran` for spatial correlation tests; `dfbeta` for influence statistics; `estat vif` for variance inflation factors for regressors; and `estat esize` for effect sizes. Additional `estat` commands following `regress` provide diagnostic tests for time-series regression. Similarly, specific postestimation `estat` commands are available following many other estimation commands.

Many of these postestimation commands are presented in this chapter and the subsequent chapter, while others are presented later in the book.

### 3.5.4 Regression subject to constraints on the parameters

A regression model can be fit subject to constraints on the parameters.

For example, to obtain the least-squares estimates subject to  $\beta_{\text{phylim}} = \beta_{\text{actlim}}$ , we define the constraint using `constraint define` and then fit the model using `cnsreg` for constrained regression with the `constraints()` option.

```
. constraint 1 phylim = actlim
. cnsreg ltotexp suppins phylim actlim totchr age female income,
> constraints(1) vce(robust)
(output omitted)
```

See exercise 2 at the end of this chapter for an example.

### 3.5.5 Hypothesis tests

The `test` command performs hypothesis tests using the Wald test procedure that uses the fitted model coefficients and VCE. We present some leading examples here, with a more extensive discussion deferred to section [11.3](#). The *F* statistic version of the Wald test is used after `regress`, whereas for many other estimators, the chi-squared version is instead used.

A common test is one of equality of coefficients. For example, consider testing that having a functional limitation has the same impact on medical expenditures as having an activity limitation. The test of  $H_0: \beta_{\text{phylim}} = \beta_{\text{actlim}}$  against  $H_a: \beta_{\text{phylim}} \neq \beta_{\text{actlim}}$  is implemented as

```
. * Wald test of equality of coefficients
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. test phylim = actlim
( 1) phylim - actlim = 0
      F( 1, 2947) =     0.27
      Prob > F =    0.6054
```

Because  $p = 0.61 > 0.05$ , we do not reject the null hypothesis at the 5% significance level. There is no statistically significant difference between the coefficients of the two variables.

Another common test is one of the joint statistical significance of a subset of the regressors. A test of the joint significance of the health-status measures is one of  $H_0: \beta_{\text{phylim}} = 0, \beta_{\text{actlim}} = 0, \beta_{\text{totchr}} = 0$  against  $H_a:$  that at least one is nonzero. This is implemented as

```

. * Joint test of statistical significance of several variables
. test phylim actlim totchr
( 1) phylim = 0
( 2) actlim = 0
( 3) totchr = 0
      F(  3,  2947) =  272.36
      Prob > F =    0.0000

```

These three variables are jointly statistically significant at the 0.05 level because  $p = 0.00 < 0.05$ .

### 3.5.6 Tables of output from several regressions

It is very useful to be able to tabulate key results from multiple regressions for both one's own analysis and final report writing.

The `estimates store` command after regression leads to results in `e()` being associated with a user-provided model name and preserved even if subsequent models are fit. Given one or more such sets of stored estimates, `estimates table` presents a table of regression coefficients (the default) and, optionally, additional results. A related command is the `estimates selected` command, which enables results to be sorted based on the variable names or coefficient magnitudes. The `estimates stats` command lists the sample size and several likelihood-based statistics.

We compare the original regression model with a variant that replaces `income` with `educyr`. The example uses several of the available options for `estimates table`.

```

. * Store and then tabulate results from multiple regressions
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. estimates store REG1
. qui regress ltotexp suppins phylim actlim totchr age female educyr, vce(robust)
. estimates store REG2

```

```
. estimates table REG1 REG2, b(%9.4f) se stats(N r2 F 11)
> stfmt(%9.2f) keep(suppins income educyr)
```

Variable	REG1	REG2
suppins	0.2556 0.0466	0.2063 0.0471
income	0.0025 0.0010	
educyr		0.0480 0.0070
N	2955	2955
r2	0.23	0.24
F	126.97	132.53
11	-4733.45	-4710.96

Legend: b/se

This table presents coefficients (`b`) and standard errors (`se`); other available options include  $t$  statistics (`t`) and  $p$ -values (`p`). The statistics given are the sample size, the  $R^2$ , the overall  $F$  statistic (based on the robust estimate of the VCE), and the log likelihood (based on the strong assumption of normal homoskedastic errors). The `keep()` option, like the `drop()` option, provides a way to tabulate results for just the key regressors of interest. Here `educyr` is a much stronger predictor than `income` because it is more highly statistically significant and  $R^2$  is higher, and there is considerable change in the coefficient of `suppins`.

### 3.5.7 Even better tables of regression output

The preceding table is very useful for model comparison but has several limitations. It would be more readable if the standard errors appeared in parentheses. It would be beneficial to be able to report a  $p$ -value for the overall  $F$  statistic. Also, some work may be needed to import the table into a table format in external software such as Excel, Word, or LaTeX.

There are several ways to proceed.

The `etable` command, introduced in an update to Stata 17, is designed specifically for tables of regression results. A richer table for the current

example is the following:

```
. etable, estimates(REG1 REG2) keep(suppins income educyr)
>      cstat(_r_b, nformat(%8.4f)) cstat(_r_se, nformat(%8.4f))
>      mstat(N) mstat(r2) mstat(F) mstat(ll) column(estimate)
>      stars(0.1 "*" 0.05 "**" 0.01 "***") showstars showstarsnote
```

	REG1	REG2
Has supp priv insurance	0.2556 *** (0.0466)	0.2063 *** (0.0471)
Annual household income/1000	0.0025 ** (0.0010)	
Years of education		0.0480 *** (0.0070)
Number of observations	2955	2955
R-squared	0.23	0.24
F statistic	126.97	132.53
Log likelihood	-4733.45	-4710.96

\*\*\* p<.01, \*\* p<.05, \* p<.1

To export this table to, for instance, a LaTeX file called `mytable.tex`, we could add the option `export(mytable.tex)`.

Several community-contributed commands provide better output than the simplest Stata commands. The Stata add-on `esttab` of [Jann \(2007\)](#) is an extension of `estimates table`. Related add-ons of [Jann \(2005, 2007\)](#) are `estout`, a richer but more complicated version of `esttab`, and `eststo`, which extends `estimates store`.

The community-contributed commands `outreg` ([Gallup 2012a](#)), `frmttable` ([Gallup 2012b](#)), and `outreg2` ([Wada 2005](#)) also create tables and generate formatted Word and LaTeX files.

Finally, one can use the more flexible version of the `table` command, introduced in Stata 17, that permits considerable formatting. For the current example, we have

```

. * Tabulate results with some formatting using the table command
. global xlist1 suppins phylim actlim totchr age female
. table (colname[suppins income educyr] result) (command),
>     command(_r_b _r_se: regress ltotexp $xlist1 income, vce(robust))
>     command(_r_b _r_se: regress ltotexp $xlist1 educyr, vce(robust))
>     nformat(%8.4f) sformat("(%s)" _r_se) style(table-reg3)
>     stars(_r_p 0.01 "***" 0.05 "**" 0.1 "*", attach(_r_b))

```

	1	2
Has supp priv insurance	0.2556*** (0.0466)	0.2063*** (0.0471)
Annual household income/1000	0.0025** (0.0010)	
Years of education		0.0480*** (0.0070)

The `collect` command, introduced in Stata 17, provides even more flexibility. Here we add measures of fit for each model and provide better labeling.

```

. * Tabulate results with even better formatting using the collect command
. capture collect clear
. collect title "Regression with income or education"
. collect notes "Heteroskedastic-robust standard errors"
. qui: collect _r_b _r_se, tag(model[(1): Income]):
>     regress ltotexp suppins $xlist1 income, vce(robust)
. qui: collect _r_b _r_se, tag(model[(2): Education]):
>     regress ltotexp suppins $xlist1 educyr, vce(robust)
. collect label values colname suppins "Supplementary insurance", modify
. collect stars _r_p 0.01 "***" 0.05 "**" 0.1 "*", attach(_r_b)
. collect style cell, nformat(%9.4f) border(right, pattern(nil))
. collect style cell result[_r_se], sformat("(%s)")
. collect style header result[N r2 F ll], level(label)
. collect label levels result r2 "R-squared", modify
. collect style header result, level(hide)
. collect style column, extraspase(1)
. collect style row stack, spacer delimiter(" x ")
. qui: collect layout (colname[suppins income educyr]#result result[N r2 F ll])
> (model)

```

```
. collect preview
```

Regression with income or education

	(1): Income	(2): Education
Supplementary insurance	0.2556*** (0.0466)	0.2063*** (0.0471)
Annual household income/1000	0.0025** (0.0010)	
Years of education		0.0480*** (0.0070)
Number of observations	2955.0000	2955.0000
R-squared	0.2289	0.2406
F statistic	126.9723	132.5337
Log likelihood	-4.73e+03	-4.71e+03

Heteroskedastic-robust standard errors

The table can be written to a file that, for example, creates a table in Microsoft Word.

```
. * Write tabulated results to a file in Microsoft Word format
. collect export mytable.docx, replace
(collection default exported to file mytable.docx)
```

The `collect` command is sufficiently flexible to warrant its own manual, *[TABLES] Stata Customizable Tables and Collected Results Reference Manual*.

### 3.5.8 Factor variables for categorical variables and interactions

Suppose we wish to add as regressors to the regression model a set of indicator variables for family size and this set of indicators interacted with income. From sections [1.3.4](#) and [2.4.7](#), the factor variables `i.famsze` form a set of indicator variables based on the nonnegative, integer-valued categorical variable `famsze`, and the factor-variables operator `c.income#i.famsze` denotes the continuous variable `income` interacted with the set of indicators.

```

. * Factor variables for sets of indicator variables and interactions
. regress ltotexp suppins phylim actlim totchr age female c.income
>     i.famsze c.income#i.famsze, vce(robust) noheader allbaselevels
note: 8.famsze#c.income omitted because of collinearity.
note: 13.famsze#c.income omitted because of collinearity.

```

ltotexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
suppins	.2393808	.0466804	5.13	0.000	.1478511	.3309104
phylim	.3053458	.0575971	5.30	0.000	.192411	.4182807
actlim	.3464812	.0631655	5.49	0.000	.2226279	.4703345
totchr	.3743755	.0187983	19.92	0.000	.3375162	.4112347
age	.00313	.0037607	0.83	0.405	-.0042438	.0105039
female	-.0725641	.0475022	-1.53	0.127	-.1657051	.0205769
income	.0028057	.0015684	1.79	0.074	-.0002695	.0058809
famsze						
1	0 (base)					
2	.0759158	.0722829	1.05	0.294	-.0658145	.2176462
3	-.2180488	.1310662	-1.66	0.096	-.4750399	.0389423
4	-.2928383	.1983967	-1.48	0.140	-.6818493	.0961727
5	.393989	.4501513	0.88	0.382	-.4886557	1.276634
6	-.3438142	.4524585	-0.76	0.447	-1.230983	.5433545
7	-1.101773	.5046005	-2.18	0.029	-2.09118	-.1123653
8	.216274	.0625337	3.46	0.001	.0936596	.3388884
10	1.482976	.2976336	4.98	0.000	.8993834	2.066568
13	-1.874285	.0712566	-26.30	0.000	-2.014003	-1.734567
famsze#c.income						
1	0 (base)					
2	-.0012899	.0020704	-0.62	0.533	-.0053495	.0027697
3	.004134	.0039464	1.05	0.295	-.003604	.0118719
4	.0160613	.0083284	1.93	0.054	-.0002688	.0323915
5	-.0251491	.017609	-1.43	0.153	-.0596764	.0093781
6	.0280329	.0227835	1.23	0.219	-.0166403	.0727062
7	-.0324118	.0279151	-1.16	0.246	-.087147	.0223234
8	0 (omitted)					
10	-.1759027	.0169673	-10.37	0.000	-.2091717	-.1426337
13	0 (omitted)					
_cons	6.748094	.3005551	22.45	0.000	6.158773	7.337414

Here there are 10 possible indicator variables for family size (1–8, 10, and 13), and the indicator for the lowest valued of these (`famsze = 1`) is the base category that is omitted from the regression. In principle, there should be as many interactions with `income` included in the regression, but those corresponding to `famsze` equal to 8 and 13 are omitted because they are not identified for these data where only one observation has `famsze` equal to 8 and only one has `famsze` equal to 13.

We can test for joint significance of the sets of indicator variables, including their interaction with income, with the following command:

```
. * Test joint significance of sets of indicator variables and interactions
. testparm i.famsze c.income#i.famsze
( 1) 2.famsze = 0
( 2) 3.famsze = 0
( 3) 4.famsze = 0
( 4) 5.famsze = 0
( 5) 6.famsze = 0
( 6) 7.famsze = 0
( 7) 8.famsze = 0
( 8) 10.famsze = 0
( 9) 13.famsze = 0
(10) 2.famsze#c.income = 0
(11) 3.famsze#c.income = 0
(12) 4.famsze#c.income = 0
(13) 5.famsze#c.income = 0
(14) 6.famsze#c.income = 0
(15) 7.famsze#c.income = 0
(16) 10.famsze#c.income = 0
F( 16,  2931) =    83.76
Prob > F =      0.0000
```

The sets of indicator variables for `famsze` are jointly statistically significant at level 0.05 because  $p = 0.00 < 0.05$ . The number of degrees of freedom is 16, so the additional two omitted variables (interaction with `income` when `famsze` equals 8 or 13) were correctly accounted for.

Calculation of the MES with respect to income or family size, which is more complicated in this model, is considered in the next section.

### 3.5.9 Average marginal effects

It becomes difficult to interpret regression results when interacted regressors are present, or more generally when models become nonlinear in the variables of interest. For example, suppose

$E(y|x, z) = \beta_1 + \beta_2 x + \beta_3 z + \beta_4 x \times z$ . Then changes in  $x$  lead to changes in  $E(y|x, z)$  through both the regressor  $x$  and the interacted regressor  $x \times z$ . Using calculus methods, we obtain the ME of  $x$  as

$\partial E(y|x, z)/\partial x = \beta_2 + \beta_4 z$ . Similarly, if a regressor enters quadratically, with  $E(y|x) = \beta_1 + \beta_2 x + \beta_3 x^2$ , then changes in  $x$  lead to changes in

$E(y|x)$  through both the regressor  $x$  and its square  $x^2$ . Using calculus methods, we obtain the ME of  $x$  as  $\partial E(y|x)/\partial x = \beta_2 + 2\beta_3x$ .

Despite the simplicity of the previous examples, there are many ways to compute such MES.

First, they vary according to what values of the regressors  $x$  and  $z$  the MES are evaluated at. Common approaches are 1) to evaluate for each sample observation and then average (called an average ME or AME); 2) to evaluate at the sample mean of the regressors; 3) to evaluate at specified values of the regressors; and 4) to use a combination of these approaches with specified values given for only some of the regressors.

Second, MES can be computed using calculus methods or by instead using the finite-difference method that changes regressors by one unit, so  $\Delta E(y|x)/\Delta x = E(y|x+1) - E(y|x)$ . For linear models such as  $E(y|x) = \beta_1 + \beta_2x$  that do not have complications such as interactions or polynomials, we obtain the ME as  $\beta_2$  using either calculus or finite-difference methods. But in nonlinear models such as the logit model with  $E(y|x) = g(\beta_1 + \beta_2x)$ , calculus and finite-difference methods lead to different results. Calculus methods are used for continuous regressors, while finite-difference methods are often used for discrete regressors.

In this section, we consider only average MES; these provide a very useful summary of the impact of each underlying variable. AMES can be computed using the `margins` postestimation command with the `dydx()` option. Such use of the `margins` command requires that factor-variable operators be used appropriately in defining the fitted model. For example, if  $x$  and  $z$  are continuous regressors, in the preceding motivating examples, we need to use, respectively, commands `regress y c.x##c.z` and `regress y c.x##c.x`.

As illustration, consider the regression model of section [3.2.4](#) with two complications. First, age enters quadratically. Because variable `age` is continuous, the regressor list will include `c.age##c.age`. Second, the continuous regressor `income` is interacted with the categorical regressor `female`, so the regressor list includes `c.income##i.female`.

For completeness, we use factor-variable notation to additionally identify the remaining regressors as either categorical indicators or continuous. This is not necessary in this specific example, because the AME following OLS regression for the remaining regressors is simply the estimated coefficient using either finite difference or calculus methods. The first three regressors are binary indicators and enter with the `i.` prefix. The regressor `totchr` was treated as a continuous regressor and enters with the `c.` prefix; if instead we use `i.totchr`, then the model would include a set of indicator variables for the various values taken by variable `totchr`. We use the `nofvlabel` option to display factor-variable level values rather than value labels. We obtain

```
. * Factor variables for model with interactions and quadratic
. regress ltotexp i.suppins i.phylim i.actlim c.totchr c.age##c.age
>      i.female##c.income, vce(robust) noheader nofvlable
```

ltotexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
1.suppins	.259794	.0467268	5.56	0.000	.1681735	.3514145
1.phylim	.3036022	.0575284	5.28	0.000	.1908023	.4164022
1.actlim	.3741712	.0631516	5.92	0.000	.2503455	.4979969
totchr	.3722821	.0186565	19.95	0.000	.335701	.4088632
age	.2832178	.087228	3.25	0.001	.1121839	.4542518
c.age#c.age	-.0018595	.0005795	-3.21	0.001	-.0029958	-.0007232
1.female	-.1819534	.0666154	-2.73	0.006	-.3125709	-.0513358
income	.0009244	.0013742	0.67	0.501	-.0017701	.0036189
female#c.income						
1	.0045633	.0020141	2.27	0.024	.000614	.0085125
_cons	-3.677626	3.267329	-1.13	0.260	-10.08411	2.728855

Variable `age` appears quadratically as  $0.283 \times \text{age} - 0.00186 \times \text{age}^2$  with ME  $0.283 - 0.00372 \times \text{age}$ , which varies with `age`. For example, the ME is 0.097 at `age` = 50. And variables `female` and `income` appear interactively as  $-0.182 \times \text{female} + 0.000924 \times \text{income} + 0.00456 \times \text{female} \times \text{income}$ . The ME of `female` is then  $-0.182 + 0.00456 \times \text{income}$ , and the ME of `income` is  $0.000924 + 0.00456 \times \text{female}$ . These MES clearly vary with the point of evaluation.

The `margins, dydx(*)` command (see section 4.5) then yields the average MES for all variables.

```

. * Average MEs in model with interactions and quadratic
. margins, dydx(*) nofvlabel
Average marginal effects                                         Number of obs = 2,955
Model VCE: Robust
Expression: Linear prediction, predict()
dy/dx wrt: 1.suppins 1.phylim 1.actlim totchr age 1.female income

```

	Delta-method					
	dy/dx	std. err.	t	P> t	[95% conf. interval]	
1.suppins	.259794	.0467268	5.56	0.000	.1681735	.3514145
1.phylim	.3036022	.0575284	5.28	0.000	.1908023	.4164022
1.actlim	.3741712	.0631516	5.92	0.000	.2503455	.4979969
totchr	.3722821	.0186565	19.95	0.000	.335701	.4088632
age	.0070988	.003857	1.84	0.066	-.0004639	.0146614
1.female	-.0784425	.0456887	-1.72	0.086	-.1680276	.0111425
income	.0035898	.0010747	3.34	0.001	.0014826	.005697

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

For the first four variables, the average MES are simply the estimated regressor coefficients. For the remaining variables that enter quadratically or interactively, we have the following average MES. Aging one year is associated with a 0.0071 increase in `ltotexp`, which corresponds to a 0.71% increase in the level of total medical expenditure. Being female is associated with a 0.0784 decrease in `ltotexp`, or a 7.84% decrease in total medical expenditure. And a \$1,000 increase in annual household income is associated with a 0.0036 increase in `ltotexp`, or a 0.36% increase in total medical expenditure. The output additionally permits statistical inference. Using a two-sided test at level 0.05, we find that the effects of `age` and `female` are statistically insignificant and that the other variables are statistically significant.

The `margins` command, and the related `marginsplot` command, is presented in much greater detail in sections [4.4–4.5](#).

### 3.5.10 Cluster-robust standard errors

As a purely illustrative example of clustering, we suppose that model errors are correlated for individuals who have the same age and are independent for those of different ages. For the current data, the variable `age` takes 26 unique values, so there are 26 clusters of varying sizes.

Cluster-robust standard errors can be obtained using the `vce(cluster clustvar)` option of the `regress` command, where here we use `vce(cluster age)` because clusters are defined by the unique values of variable `age`. We obtain

```
. * Cluster-robust standard errors
. regress ltotexp suppins phylim actlim totchr age female income, vce(cluster age)
Linear regression
Number of obs      =      2,955
F(7, 25)          =     178.77
Prob > F          =     0.0000
R-squared          =     0.2289
Root MSE           =     1.2023
(Std. err. adjusted for 26 clusters in age)
```

ltotexp	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
suppins	.2556428	.0546516	4.68	0.000	.1430856 .3681999
phylim	.3020598	.0634415	4.76	0.000	.1713995 .43272
actlim	.3560054	.06361	5.60	0.000	.2249981 .4870127
totchr	.3758201	.0153014	24.56	0.000	.3443064 .4073339
age	.0038016	.0063709	0.60	0.556	-.0093195 .0169228
female	-.0843275	.0469837	-1.79	0.085	-.1810923 .0124372
income	.0025498	.0013169	1.94	0.064	-.0001623 .0052619
_cons	6.703737	.4943524	13.56	0.000	5.6856 7.721875

For this somewhat contrived example, we expect little difference from the heteroskedastic-robust standard errors given earlier. In fact, for several variables, there is little difference; for several variables, the cluster-robust standard errors are larger; and for `totchr`, the cluster-robust standard errors are smaller.

### 3.5.11 Bootstrap standard errors

Bootstrap methods, introduced in section [3.4.7](#) and detailed in chapter [12](#), are resampling methods that are most often used to obtain standard errors when it is difficult to obtain standard errors by other methods.

A key ingredient is the number of bootstrap resamples, denoted  $B$ , to use with a tradeoff between computational time and precision. Unless otherwise stated, we use  $B = 400$  or  $B = 999$ ; see section [12.3.4](#). To ensure that future execution of the same command leads to the same bootstrap samples, and

hence the same bootstrap standard errors, we set the seed for the random-number generator used to determine the bootstrap resamples.

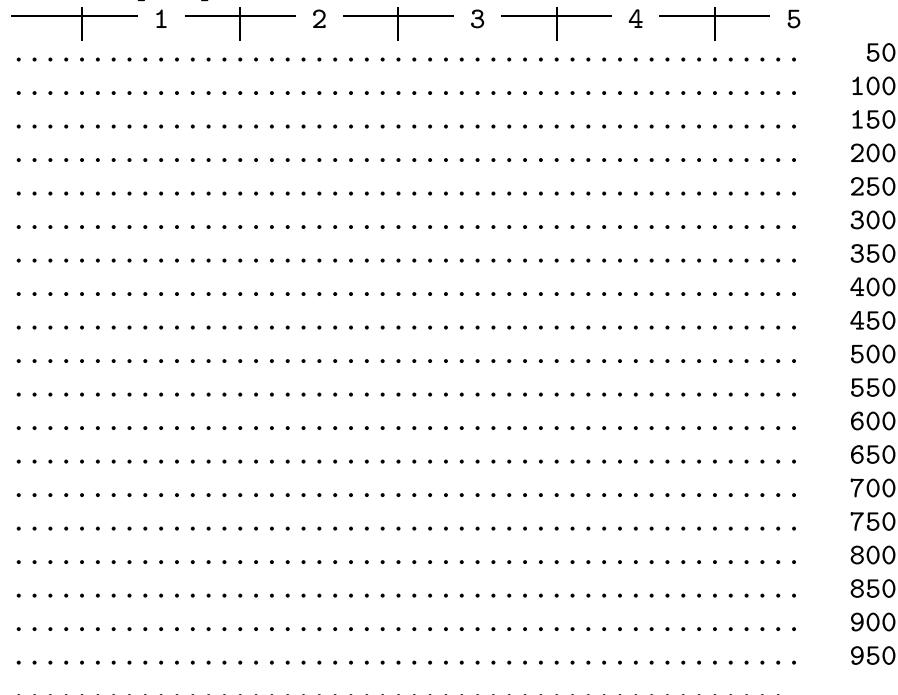
There are many ways to bootstrap. We first consider a bootstrap to obtain heteroskedastic-robust standard errors. This can be obtained using the `vce(bootstrap)` option of the `regress` command. We obtain

```

. * Bootstrap to give heteroskedastic-robust standard errors
. regress ltotexp suppins phylim actlim totchr age female income,
> vce(bootstrap, reps(999) seed(10101))
(running regress on estimation sample)

```

Bootstrap replications (999)



Linear regression

Number of obs	=	2,955
Replications	=	999
Wald chi2(7)	=	958.51
Prob > chi2	=	0.0000
R-squared	=	0.2289
Adj R-squared	=	0.2271
Root MSE	=	1.2023

ltotexp	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
suppins	.2556428	.0469263	5.45	0.000	.163669	.3476165
phylim	.3020598	.058277	5.18	0.000	.187839	.4162806
actlim	.3560054	.0653862	5.44	0.000	.2278508	.48416
totchr	.3758201	.0184238	20.40	0.000	.3397102	.4119301
age	.0038016	.003648	1.04	0.297	-.0033483	.0109516
female	-.0843275	.0457723	-1.84	0.065	-.1740396	.0053845
income	.0025498	.001072	2.38	0.017	.0004488	.0046508
_cons	6.703737	.2768602	24.21	0.000	6.161101	7.246374

These bootstrap heteroskedastic-robust standard errors are within 3% of the heteroskedastic-robust standard errors obtained using the `vce(robust)`

option. The biggest relative difference is for `actlim`, which has a standard error of 0.06539 compared with 0.06361.

The preceding bootstrap standard errors are asymptotically equivalent to the preceding heteroskedastic–robust standard errors. Part of the observed difference is due to using only 999 bootstrap replications, a difference that diminishes as the number of bootstrap replications increases. And part of the difference is due to finite degrees-of-freedom corrections.

We next obtain cluster–robust bootstrap standard errors using the `vce(bootstrap, cluster())` option of the `regress` command. We obtain

```
. * Bootstrap to give cluster--robust standard errors
. regress ltotexp suppins phylim actlim totchr age female income,
>      vce(bootstrap, cluster(age) nodots reps(999) seed(10101))
Linear regression                                         Number of obs = 2,955
                                                               Replications = 999
                                                               Wald chi2(7) = 1110.58
                                                               Prob > chi2 = 0.0000
                                                               R-squared = 0.2289
                                                               Adj R-squared = 0.2271
                                                               Root MSE = 1.2023
(Replications based on 26 clusters in age)
```

ltotexp	Observed	Bootstrap	Normal-based		
	coefficient	std. err.	z	P> z	[95% conf. interval]
suppins	.2556428	.0544001	4.70	0.000	.1490206 .3622649
phylim	.3020598	.0613503	4.92	0.000	.1818154 .4223042
actlim	.3560054	.0646722	5.50	0.000	.2292502 .4827606
totchr	.3758201	.0159496	23.56	0.000	.3445595 .4070808
age	.0038016	.0066821	0.57	0.569	-.009295 .0168983
female	-.0843275	.0475312	-1.77	0.076	-.177487 .0088319
income	.0025498	.001266	2.01	0.044	.0000685 .0050311
_cons	6.703737	.5185035	12.93	0.000	5.687489 7.719986

These bootstrap cluster–robust standard errors are within 4% of the cluster–robust standard errors obtained earlier.

In this example, we have used the `nodots` option to suppress the output of a dot for each bootstrap replication. More generally, the command `set dots off` will suppress dots in all commands that by default produce dots.

Sometimes, one is interested in specific nonlinear functions of estimated coefficients (and data) that may have economic meaning and significance, for example, the ratio of two coefficients or the elasticity of the dependent variable with respect to a particular regressor. Exact confidence intervals for such a function are generally difficult to compute, and one often settles for an approximation based on the delta method. In such instances, the bootstrap method provides an alternative.

To illustrate this, we consider inference on the ratio of the slope parameters for `phylim` and `actlim`. We can no longer use the `vce(bootstrap)` option of the `regress` command and instead need to use the `bootstrap` prefix. Here the quantity being bootstrapped is denoted `_b[suppins]/_b[phylim]`. We obtain

```
. * Bootstrap to compute confidence interval for a ratio of two coefficients
. bootstrap _b[suppins]/_b[phylim], reps(999) seed(10101) nodots:
>      regress ltotexp suppins phylim actlim totchr age female income
Linear regression                                         Number of obs = 2,955
                                                               Replications = 999
Command: regress ltotexp suppins phylim actlim totchr age female income
         _bs_1: _b[suppins]/_b[phylim]
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]
_bs_1	.8463317	.262695	3.22	0.001	.3314588 1.361204

Although the sample estimate of the ratio is close to 1, the confidence interval is quite wide. A similar approach can be used to construct bootstrap confidence intervals for other nonlinear functions of parameters. However, care should be exercised to ensure that the target function is differentiable and well behaved. For example, if the function is a ratio, then the denominator needs to be bounded away from 0 so that the moments exist.

### 3.5.12 OLS regression to test mean and difference in mean

The methods presented in an introductory statistics course for inference on the mean of a single variable, using a one-sample  $t$  test, or on the difference in mean, using a two-sample  $t$  test, are implemented in Stata using the `ttest` command.

These tests can also be implemented using OLS regression. OLS regression has many advantages over the `ttest` command. If errors are clustered, for example, we can perform valid inference on the mean by using command `regress y, vce(cluster id_clu)`. We can add control variables, as already done in the preceding analysis. With many categories of health insurance, we can simply regress on indicator variables for the various categories (with one category omitted). And extension to difference in differences is straightforward; see section [4.8](#).

### **ttest command**

For a one-sample  $t$  test on the mean  $\mu$  of a single variable, the command syntax is

```
ttest varname == # [if] [in] [, level(#) ]
```

Inference is based on assuming that

$$t = \frac{\bar{y} - \mu}{s/\sqrt{N}} \sim t(N - 1)$$

For a two-sample  $t$  test of the difference in means  $\mu_1 - \mu_0$  of two variables  $y_1$  and  $y_0$ , the command syntax is

```
ttest varname [if] [in], by(groupvar) [options]
```

Inference for the two-sample test varies with whether the variances  $\sigma_1^2$  and  $\sigma_0^2$  are assumed to be equal, and the norm in economics is to assume unequal variances. Then the `ttest` command with option `unequal` bases inference on assuming that

$$t = \frac{(\bar{y}_1 - \bar{y}_0) - (\mu_1 - \mu_0)}{\sqrt{s_1^2/N_1 + s_2^2/N_0}} \sim t(v)$$

where there are several different formulas for  $v$ , including  $v = N_1 + N_0 - 2$

For completeness, we note that the `ttest` command can also be used to test equality of means of two variables  $y_1$  and  $y_2$ , where the variables may be paired (with  $y_{1i}$  and  $y_{2i}$  data for the same individual) or unpaired.

### One-sample t test

We first consider inference on the mean  $\mu$  of a single variable, using the `mean` command that gives a confidence interval for  $\mu$  in addition to the `ttest` command. For the level of health expenditure, and a test that  $\mu = 7500$ , we obtain

```
. * Test of mean using ttest
. ttest totexp = 7500
```

One-sample t test

Variable	Obs	Mean	Std. err.	Std. dev.	[95% conf. interval]
totexp	3,064	7030.889	214.1287	11852.75	6611.039 7450.74

mean = mean(totexp)	t = -2.1908
H0: mean = 7500	Degrees of freedom = 3063
Ha: mean < 7500	Ha: mean != 7500
Pr(T < t) = 0.0143	Pr( T  >  t ) = 0.0285
	Ha: mean > 7500
	Pr(T > t) = 0.9857

Identical results are obtained by OLS regression of `totexp` on just an intercept with default standard errors (or if option `vce(robust)` is used). We have

```
. * Test of mean using regress
. regress totexp, noheader
```

totexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
_cons	7030.889	214.1287	32.83	0.000	6611.039 7450.74

```
. test _cons=7500
( 1) _cons = 7500
      F( 1, 3063) =     4.80
      Prob > F =    0.0285
```

The `ttest` postestimation command reports an  $F(1, N - 1)$  statistic that is the square of a  $t(N - 1)$  statistic. Here  $(-2.1908)^2 = 4.80$  and both tests yield  $p = 0.0285$ .

### Two-sample t test

Next consider the difference in mean health expenditure by whether the person has supplemental health insurance. The `ttest` command yields

```
. * Test of difference in means using ttest
. ttest totexp, by(suppins) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf. interval]
No	1,283	6420.058	312.6458	11198.66	5806.704
Yes	1,781	7470.921	291.1413	12286.72	6899.907
Combined	3,064	7030.889	214.1287	11852.75	6611.039
diff		-1050.864	427.2127		-1888.535
					-213.1925
		diff = mean(No) - mean(Yes)			t = -2.4598
H0:	diff = 0				Satterthwaite's degrees of freedom = 2899.24
Ha:	diff < 0				Ha: diff != 0
		Pr(T < t) = 0.0070		Pr( T  >  t ) = 0.0140	Ha: diff > 0
					Pr(T > t) = 0.9930

OLS regression of `totexp` on `suppins` with heteroskedastic-robust standard errors yields almost identical results.

```
. * Test of difference in means using regress
. regress totexp suppins, vce(robust)
```

Linear regression	Number of obs	=	3,064
	F(1, 3062)	=	6.05
	Prob > F	=	0.0140
	R-squared	=	0.0019
	Root MSE	=	11843

totexp	Coefficient	Robust		t	P> t	[95% conf. interval]
		std. err.				
suppins	1050.864	427.2072		2.46	0.014	213.2218
_cons	6420.058	312.626		20.54	0.000	5807.08
						7033.036

We obtain the same point estimate of 1050.864. The standard error differs slightly (427.2072 compared with 427.2127) because of slightly different ways that degrees-of-freedom adjustments are made. The `ttest` command uses the  $t(2899.24)$  distribution, using Satterthwaite's formula, whereas `regress` uses  $v = N_1 + N_0 - 2 = 3062$ . These degrees-of-freedom differences make very little difference here; with few observations, the differences are larger.

## 3.6 Specification analysis

The fitted model in section [3.5.2](#) has  $R^2 = 0.23$ , which is reasonable for cross-sectional data, and most regressors are highly statistically significant with the expected coefficient signs. Therefore, it is tempting to begin interpreting the results.

However, before we do so, it is useful to subject this regression to some additional scrutiny because a badly misspecified model may lead to erroneous inferences. Stata also presents the user with an impressive and bewildering menu of choices of diagnostic checks for the currently fitted regression; see [R] **regress postestimation**. Some are specific to OLS regression, whereas others apply to most regression models. Some are visual aids such as plots of residuals against fitted values. Some are diagnostic statistics such as influence statistics that indicate the relative importance of individual observations. And some are formal tests that test for the failure of one or more assumptions of the model.

In this section, we focus on detecting and controlling for influential or outlying observations using model diagnostics, data transformation, and estimators other than OLS. In the subsequent section, we present several specification tests, with the notable exception of testing for regressor exogeneity, which is deferred to section [7.4.6](#).

### 3.6.1 Robust regression

The topic of robust regression (as distinct from robust variance estimation of regression coefficients) deals with ways to mitigate the limitations of standard regression analysis of data with “outliers”, or influential observations, that can potentially distort statistical inference. In essence, outliers are observations that are generated by a chance mechanism that is different from that which is assumed to generate most of the sample. That is, the available sample is a mixture of draws from more than one population, with extreme observations or outliers contributing a relatively small proportion of the sample.

Despite this fact, these few observations may create significant distortions. This provides motivation for detection or elimination, or both, of such observations. This task is challenging because outliers and extreme observations may be hard to differentiate in a small sample. Extreme observations, by definition, are low-probability events, but they are also draws from the same distribution that is assumed to generate the remaining observations in the sample. Hence, the case for eliminating such observations is weaker.

In the simple regression  $y = \mathbf{x}'\boldsymbol{\beta} + u$ , the  $y$ -outliers can be generated by either  $\mathbf{x}$ -outliers (the signal component) or  $u$ -outliers (the noise component), or both. Closely related to the notion of  $u$ -outliers is the idea of fat-tailed or heavy-tailed distributions that could generate samples with a higher proportion of extreme  $y$ -observations—relative to some benchmark distribution such as the Gaussian.

Outliers may result from measurement errors due to equipment malfunction, data coding errors, or data generated under unique circumstances. Because outliers are likely to distort regression results more severely when the sample is small, detection and elimination are more feasible in such cases. Distinguishing between outliers and heavy-tailed distributions may be difficult even in a large sample. Often, the objective is to make inference robust to the presence of such observations; see [Huber and Ronchetti \(2009\)](#).

Categorizing an observation or a group of observations as outliers, often on the basis of regression residuals, depends on the assumed benchmark error distribution or the functional form of the regression. Specifying that functional form involves the choice of transformation. Hence, the topic of robust regression is related to choice of variable transformations in regression.

Transformations involve rewriting the regression model in terms of rescaled variables, which in turn changes the scale parameter of the error distribution. The use of a “wrong” functional form may frequently lead to residuals that suggest outliers, but the same conclusion may not apply to the regression on transformed variables. Hence, outliers are to be considered in

the context of the maintained functional form; therefore, the selection of functional form is a step that should precede outlier detection.

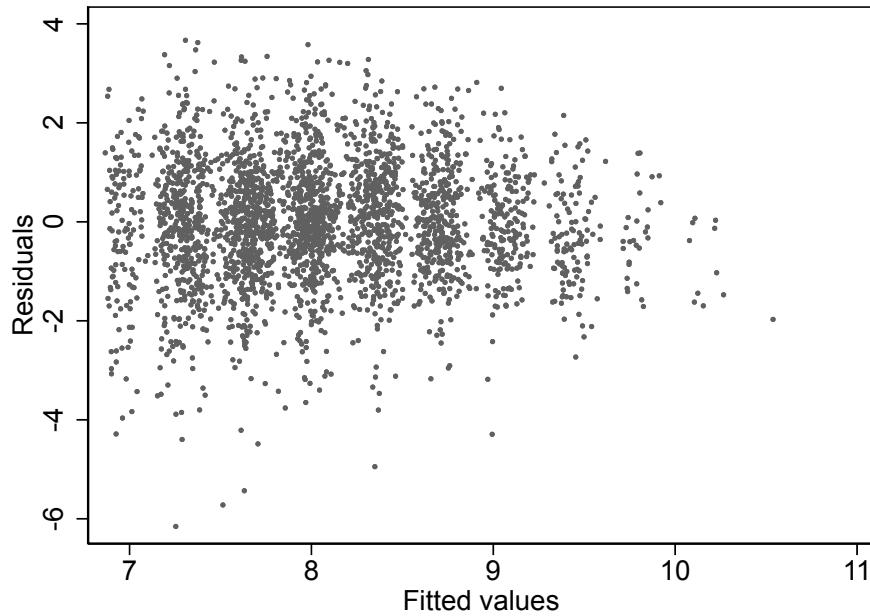
### 3.6.2 Residual diagnostic plots

Diagnostic plots are used less in microeconomics than in some other branches of statistics for several reasons. First, economic theory and previous research provide a lot of guidance as to the likely key regressors and functional form for a model. Studies rely on this and shy away from excessive data mining; see chapter 28 for data-mining methods. Second, microeconomic studies typically use large datasets and regressions with many variables. Many variables potentially lead to many diagnostic plots, and many observations make it less likely that any single observation will be very influential, unless data for that observation are seriously miscoded.

We consider various residual plots that can aid in outlier detection, where an outlier is an observation poorly predicted by the model. One way to do this is to plot actual values against fitted values of the dependent variable. The postestimation command `rvfplot` gives a transformation of this, plotting the residuals  $\hat{u}_i = y_i - \hat{y}_i$  against the fitted values  $\hat{y}_i = \mathbf{x}'_i \hat{\beta}$ .

We have

```
. * Plot of residuals against fitted values
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. rvfplot, msizen(tiny) scale(1.2)
```



**Figure 3.2.** Residuals plotted against fitted values after OLS regression

Figure 3.2 does not indicate any extreme outliers, though the three observations with a residual less than  $-5$  may be worth investigating. To do so, we need to generate  $\hat{u}$  by using the `predict` command, detailed in section 4.2, and we need to list some details on those observations with  $\hat{u} < -5$ . We have

```
. * Details on the outlier residuals
. predict uhat, residual
(109 missing values generated)
. predict yhat, xb
. list totexp ltotexp yhat uhat if uhat < -5, clean
      totexp      ltotexp        yhat        uhat
    1.          3     1.098612    7.254341   -6.155728
    2.          6     1.791759    7.513358   -5.721598
    3.          9     2.197225    7.631211   -5.433987
```

The three outlying residuals are for three observations with the very smallest total annual medical expenditures of, respectively, \$3, \$6, and \$9. The model evidently greatly overpredicts for these observations, with the predicted logarithm of total expenditures (`yhat`) much greater than `ltotexp`.

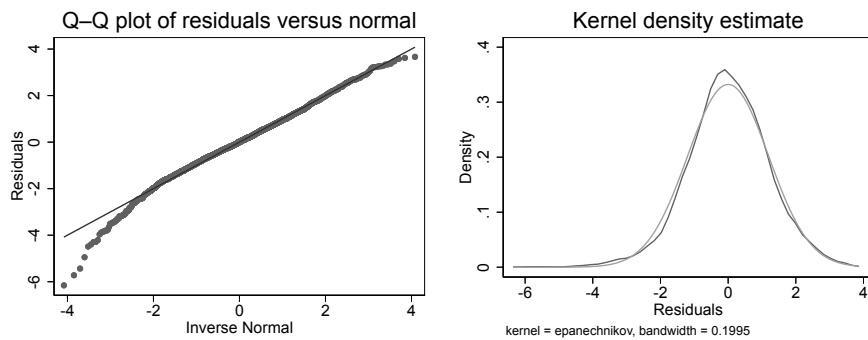
Stata provides several other residual plots. The `rvppplot` postestimation command plots residuals against an individual regressor. The `avplot` command provides an added-variable plot, or partial regression plot, that is a useful visual aid to outlier detection. Other commands give component-plus-residual plots that aid detection of nonlinearities and leverage plots. For details and additional references, see [R] **regress postestimation diagnostic plots**.

Another useful plot is a Q–Q plot, or quantile–quantile plot, that plots the quantiles of one variable against those of another variable. If the variables have the same distribution aside from centering and scaling, then we expect a linear relationship between the quantiles.

The `qqplot` command plots the quantiles of one variable against the quantiles of a second variable. The `qnorm` command instead plots the quantiles of one variable against the quantiles of a normal distribution with mean and variance of those of the first variable.

The following example creates a Q–Q plot of residuals against the normal distribution and a plot of the kernel density estimate of the residuals compared with the normal distribution, where residuals come from OLS regression of the log-linear model.

```
. * Quantile-quantile plot of fitted against actual values
. local endash = ustrunescape("\u2013")
. qnorm uhat, msizes(small) title("Q`endash`Q plot of residuals versus normal")
>     name(graph1, replace)
. kdensity uhat, normal legend(off) name(graph2, replace)
. graph combine graph1 graph2, iscale(1.2) ysize(2.5) xsize(6.0)
```



**Figure 3.3.** Q–Q plot of residuals against normal and kernel density estimate

The first panel of figure 3.3 indicates that transforming to logs has led to OLS residuals that are normally distributed, aside from the smallest residuals. As already seen, these smallest residuals correspond to unusually small medical expenditures. The second panel also suggests that residuals are approximately normal but does not so clearly show the departure from normality for the smallest residuals.

### 3.6.3 Influential observations

Some observations may have unusual influence in determining parameter estimates and resulting model predictions.

Influential observations can be detected using one of several measures that are large if the residual is large, the leverage measure is large, or both. The leverage measure of the  $i$ th observation, denoted by  $h_i$ , equals the  $i$ th diagonal entry in the so-called hat matrix  $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$ . If  $h_i$  is large, then  $y_i$  has a big influence on its OLS prediction  $\hat{y}_i$  because  $\hat{\mathbf{y}} = \mathbf{Hy}$ . Different measures, including  $h_i$ , can be obtained by using different options of `predict`.

A commonly used measure is  $\text{DFITS}_i$ , which can be shown to equal the (scaled) difference between predictions of  $y_i$  with and without the  $i$ th observation in the OLS regression (so DFITS means “difference in fits”). Large absolute values of DFITS indicate an influential data point. One can plot DFITS and investigate further observations with outlying values of DFITS. A rule of thumb is that observations with  $|\text{DFITS}| > 2\sqrt{K/N}$  may be worthy of further investigation, though for large datasets this rule can suggest that many observations are influential.

The `dfits` option of `predict` can be used after `regress` provided that regression is with default standard errors because the underlying theory presumes homoskedastic errors. We have

```

. * Compute dfits that combines outliers and leverage
. qui regress ltotexp suppins phylim actlim totchr age female income
. predict dfits, dfits
(109 missing values generated)
. scalar threshold = 2*sqrt((e(df_m)+1)/e(N))
. display "dfits threshold = " %6.3f threshold
dfits threshold = 0.104
. tabstat dfits, statistics(min p1 p5 p95 p99 max) format(%9.3f)
> columns(statistics)



| Variable | Min    | p1     | p5     | p95   | p99   | Max   |
|----------|--------|--------|--------|-------|-------|-------|
| dfits    | -0.421 | -0.147 | -0.083 | 0.085 | 0.127 | 0.221 |



. list dfits totexp ltotexp yhat uhat if abs(dfits) > 2*threshold & e(sample),
> clean

      dfits    totexp    ltotexp        yhat        uhat
1.   -.2319179       3    1.098612    7.254341   -6.155728
2.   -.3002994       6    1.791759    7.513358   -5.721598
3.   -.2765266       9    2.197225    7.631211   -5.433987
10.   -.2170063      30    3.401197    8.348724   -4.947527
42.   -.2612321     103    4.634729    7.57982   -2.945091
44.   -.4212185     110    4.70048    8.993904   -4.293423
108.   -.2326284     228    5.429346    7.971406   -2.54206
114.   -.2447627     239    5.476463    7.946239   -2.469776
137.   -.2177336     283    5.645447    7.929719   -2.284273
211.   -.211344      415    6.028278    8.028338   -2.00006
2925.   .2207284    62346   11.04045    8.660131   2.380323

```

Here over 2% of the sample has  $|DFITS|$  greater than the suggested threshold of 0.104. But only 11 observations have  $|DFITS|$  greater than two times the threshold. These correspond to observations with relatively low expenditures, or in one case, relatively high expenditures. We conclude that no observation has unusual influence.

### 3.6.4 Stata's robust regression

If one wants to apply a variant of the robust regression estimator without first discarding the troublesome observations, then one option is to use the `rreg` command, whose syntax is similar to that of the ordinary `regress` command,

```
rreg depvar [indepvars] [if] [in] [, options]
```

In this case, the outliers, or observations that correspond to large residuals, will get reduced weight; when the weight is close to zero, the observation is effectively dropped from the sample.

Briefly, the theory behind the `rreg` command is as follows. Let  $r_i$  denote the residual  $(y_i - \mathbf{x}'_i \boldsymbol{\beta})$  and  $r_i/\sigma$  denote the scaled residual obtained by scaling the residuals by a robust estimate of the standard deviation (implicitly assuming homoskedasticity). Define the objective function for robust regression as follows,

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^N q_i \left( \frac{r_i}{\sigma} \right) = \sum_{i=1}^N q_i \left( \frac{y_i - \mathbf{x}'_i \boldsymbol{\beta}}{\sigma} \right)$$

where  $q_i(\cdot)$  is a weighting function. OLS is the special case in which the weighting function is the same quadratic function for all observations. Robust regression instead downweights this quadratic function for large residuals  $y_i - \mathbf{x}'_i \boldsymbol{\beta}$ . The first-order condition for the optimum is

$$\sum_{i=1}^N \frac{\partial q_i(r_i/\sigma)}{\partial \boldsymbol{\beta}'} \mathbf{x}_i = \mathbf{0}$$

where  $\partial q_i/\partial \boldsymbol{\beta}'$  is an observation-specific weight.

Implementation of the `rreg` command begins with OLS estimation of the regression. Observation-specific weights are calculated based on the absolute Studentized residuals, and OLS regression is run again using the weights. Estimation is iterative because the first-order conditions are solved using revised weights in each round, until convergence is achieved to satisfy the tolerance criterion. The choice of weights has an important role in the speed of convergence. The specifics of the weights are covered in the Stata manual entry for the `rreg` command.

### 3.6.5 Median regression

Another approach for handling outliers and departures from normality is to use a method that, unlike OLS and MLE assuming normality, is not based on the quadratic loss function.

Median regression, or least-absolute-deviations regression, is a special case of quantile regression that is based on minimizing the sum of absolute deviations rather than the sum of squared deviations. Then  $\beta$  minimizes

$$Q(\beta) = \sum_{i=1}^N |y_i - \mathbf{x}'_i \beta|$$

This objective function is less sensitive to extreme observations. The median estimator, a member of the class of  $m$  estimators, is implemented in Stata as a special (default) case of the quantile regression `qreg` command. A more extensive coverage of this estimator is given in chapter [15](#).

The median regression estimator offers several advantages over OLS. Its target parameter is the conditional median of  $y$ , which is less sensitive than the conditional mean to skewness and excess kurtosis. It is a semiparametric estimator and is known to be less sensitive to departures from normality and hence to the presence of outliers. It also does not require homoskedastic errors.

### 3.6.6 Robust and median regression example

First, we use the `rreg` command to implement the Huber-style robust regression using the expenditure data.

```

. * Robust regression as a check on fat tails
. qui use mus203mepsmedexp, replace
. rreg ltotexp suppins phyylim actlim totchr age female income, genwt(w)
    Huber iteration 1: maximum difference in weights = .75846426
    Huber iteration 2: maximum difference in weights = .05754505
    Huber iteration 3: maximum difference in weights = .0112007
Biweight iteration 4: maximum difference in weights = .29408276
Biweight iteration 5: maximum difference in weights = .0092795
Robust regression
                                         Number of obs      =      2,955
                                         F(  7,     2947) =     119.22
                                         Prob > F        =     0.0000

```

ltotexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
suppins	.2469393	.0451418	5.47	0.000	.1584266 .335452
phyylim	.330684	.0556342	5.94	0.000	.2215982 .4397699
actlim	.3533665	.0606545	5.83	0.000	.2344371 .4722958
totchr	.3410272	.0179905	18.96	0.000	.3057521 .3763024
age	.0056581	.0035703	1.58	0.113	-.0013425 .0126587
female	-.0843941	.0444756	-1.90	0.058	-.1716005 .0028123
income	.0026742	.0009955	2.69	0.007	.0007223 .0046261
_cons	6.643979	.2702664	24.58	0.000	6.114049 7.173909

```
. estimates store ROB_DEF
```

The robust regression method necessarily presumes model errors are homoskedastic, so there is no heteroskedastic–robust option.

The option `genwt(w)` saves in variable `w` the weights used in the `rreg` procedure. Summary statistics for the weights follow:

```

. * Weights used for robust regression
. sum w, detail
      Robust Regression Weight

```

---

	Percentiles	Smallest		
1%	.331044	0	Obs	2,955
5%	.6070053	0	Sum of wgt.	2,955
10%	.7331903	0	Mean	.9040597
25%	.8774092	.0079872	Std. dev.	.1400978
50%	.9598379		Variance	.0196274
		Largest	Skewness	-2.657079
75%	.9911338	1	Kurtosis	11.72245
90%	.9983704	1		
95%	.9995538	1		
99%	.9999782	1		

```

. drop w

```

Regarding the weights, 75% are between 0.88 and 1.0, and only 1% have weight less than 0.33. This suggests that there was little need for robust regression here.

Next, we obtain median regression estimates, the default for the `qreg` command, with heteroskedastic–robust standard errors reported.

```

. * Median regression with default standard errors
. qreg ltotexp suppins phylim actlim totchr age female income, nolog vce(robust)
Median regression                                         Number of obs =      2,955
Raw sum of deviations  1555.48 (about 8.111928)
Min sum of deviations 1368.373                           Pseudo R2      =     0.1203

```

---

ltotexp	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
suppins	.3019549	.0534219	5.65	0.000	.1972069 .4067028
phylim	.323097	.0657772	4.91	0.000	.1941231 .4520709
actlim	.3199463	.0730748	4.38	0.000	.1766635 .4632291
totchr	.3244109	.0208302	15.57	0.000	.2835676 .3652541
age	.0049243	.0041258	1.19	0.233	-.0031654 .013014
female	-.114351	.0533723	-2.14	0.032	-.2190017 -.0097003
income	.0029569	.0006201	4.77	0.000	.001741 .0041728
_cons	6.697871	.3133837	21.37	0.000	6.083398 7.312344

```

. estimates store MED_ROB

```

In this example, we have used the `nolog` option to suppress the iteration log that is produced by nonlinear estimation commands such as `qreg`. More

generally, the command `set iterlog off` will suppress the iteration log in all commands that by default produce an iteration log.

Finally, to facilitate comparison, we present a table with results from the `regress`, `qreg`, and `rreg` commands. The first three columns report default standard errors, while the last two columns report heteroskedastic–robust standard errors that are available for OLS and median regression but not for robust regression.

```
. * Compare OLS, robust regression, and median regression
. qui regress ltotexp suppins phyylim actlim totchr age female income
. estimates store OLS_DEF
. qui regress ltotexp suppins phyylim actlim totchr age female income, vce(robust)
. estimates store OLS_ROB
. qui qreg ltotexp suppins phyylim actlim totchr age female income, nolog
. estimates store MED_DEF
. estimates table OLS_DEF MED_DEF ROB_DEF OLS_ROB MED_ROB, b(%10.4f) se stats(N)
```

Variable	OLS_DEF	MED_DEF	ROB_DEF	OLS_ROB	MED_ROB
suppins	0.2556	0.3020	0.2469	0.2556	0.3020
	0.0462	0.0541	0.0451	0.0466	0.0534
phyylim	0.3021	0.3231	0.3307	0.3021	0.3231
	0.0570	0.0666	0.0556	0.0577	0.0658
actlim	0.3560	0.3199	0.3534	0.3560	0.3199
	0.0621	0.0726	0.0607	0.0634	0.0731
totchr	0.3758	0.3244	0.3410	0.3758	0.3244
	0.0184	0.0215	0.0180	0.0187	0.0208
age	0.0038	0.0049	0.0057	0.0038	0.0049
	0.0037	0.0043	0.0036	0.0037	0.0041
female	-0.0843	-0.1144	-0.0844	-0.0843	-0.1144
	0.0455	0.0533	0.0445	0.0457	0.0534
income	0.0025	0.0030	0.0027	0.0025	0.0030
	0.0010	0.0012	0.0010	0.0010	0.0006
_cons	6.7037	6.6979	6.6440	6.7037	6.6979
	0.2768	0.3237	0.2703	0.2826	0.3134
N	2955	2955	2955	2955	2955

Legend: b/se

In general, the OLS and median regression parameter estimates are not directly comparable, because OLS models the conditional mean while median regression models the conditional median. Here the use of the logarithmic transformation applied to `totexp` makes the distribution more symmetric, in

which case the median and mean are similar, so the two estimates are similar. And the difference between OLS and robust regression estimates is not great here, due again to the logarithmic transformation bringing in outliers and also because the leverage of the outliers or extreme observations, or both, is small given a sample that is quite large.

It is interesting to repeat the exercise using the highly skewed variable `totexp` in place of `ln(totexp)` as the dependent variable. We expect that the differences between the three estimators in this case will be larger. We obtain

```
. * OLS, robust, and median regression for positive level of total expenditures
. replace totexp = . if totexp <= 0
(109 real changes made, 109 to missing)
. qui rreg totexp suppins phylim actlim totchr age female income, genwt(w)
. estimates store ROB_LEVEL
. qui qreg totexp suppins phylim actlim totchr age female income, vce(robust)
. estimates store MED_LEVEL
. qui regress totexp suppins phylim actlim totchr age female income, vce(robust)
. estimates store OLS_LEVEL
. estimates table OLS_LEVEL MED_LEVEL ROB_LEVEL, b(%10.4f) se stats(N)
```

Variable	OLS_LEVEL	MED_LEVEL	ROB_LEVEL
suppins	724.8632	784.3740	725.8082
	427.3045	131.0828	115.1799
phylim	2389.0186	901.9890	638.8099
	544.3493	245.2875	141.9514
actlim	3900.4908	1539.7285	673.1755
	705.2244	349.2155	154.7606
totchr	1844.3769	1114.7952	823.2710
	186.8938	70.5389	45.9029
age	-85.3626	-1.8305	17.4398
	37.8187	11.3550	9.1098
female	-1383.2898	-131.1867	-77.9987
	432.4759	149.8859	113.4801
income	6.4689	7.1615	6.0682
	8.5707	3.3079	2.5400
_cons	8358.9539	512.7608	-230.2268
	2847.8024	848.4860	689.5881
N	2955	2955	2955

Legend: b/se

The differences across estimation methods in the estimates is much greater, especially for the coefficients of `phylim`, `actlim`, and `female`. The linear version has heteroskedastic and skewed errors, and the three estimators show different sensitivity to these features. Note that it would not be correct to attribute these differences to outliers in the regressors. As was seen in the previous table, there was very little variation between estimators in the case of regression with a log-transformed dependent variable.

## 3.7 Specification tests

There are several ways in which a regression may be misspecified. Omitted variables and misspecified functional form are two important reasons for a regression either not fitting the data well or generating poor predictions in the postsample period. This provides motivation for statistical tests of functional forms.

Linearity of the conditional mean function implies  $E(y|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta} = \sum_{j=1}^K \beta_j x_{ij}$ , which is a strong assumption. Specifically, the MES  $\partial E(y|\mathbf{x})/\partial x_j = \beta_j$  are constant and independent of the level of  $x_j$ . If we would prefer a more flexible parametric functional form, the conditional mean function can be modified, or else we can test for the presence of nonlinearity of  $E(y|\mathbf{x})$ .

1. For a selected subset of regressors, we can generate nonlinearity by adding polynomial terms such as  $x_{ij}^2$  and  $x_{ij}^3$  to the conditional mean function. Using standard  $t$  tests or  $F$  tests, we can then check whether the coefficients of these additional variables are statistically significant.
2. We can generate a set of interaction variables such as  $x_{ij}x_{ik}$  and add them to the conditional mean function. Again, using standard  $t$  tests or  $F$  tests, we can check whether these variables generate a significant nonlinearity in the conditional mean function.
3. Using an initial specification of the regression, we first fit the model. Then using fitted values  $\mathbf{x}'\hat{\boldsymbol{\beta}}$ , we generate new variables  $(\mathbf{x}'\hat{\boldsymbol{\beta}})^2$ ,  $(\mathbf{x}'\hat{\boldsymbol{\beta}})^3, \dots, (\mathbf{x}'\hat{\boldsymbol{\beta}})^p$ . These  $p$  generated regressors are added to the original regression, which is reestimated, and an  $F$  test of the joint significance of the generated regressors is applied. Rejection of the null hypothesis that the added variables are jointly insignificant is evidence of departure from linearity. The test is known as the RESET test ([Cameron and Trivedi 2005](#), 277–278).

The first two of these three tests are less mechanical than the last because they motivate one to consider how the nonlinearity may arise and possible economic justification for any specific nonlinearity. For example, we may have reason to expect that the ME with respect to  $x_{ij}$  may depend upon  $x_{ik}$ .

Additionally, formal model-specification tests exist whereby rejection of the null hypothesis means that key assumptions of the model are rejected. Such tests have two limitations. First, a test for the failure of a specific model assumption may not be robust with respect to the failure of another assumption that is not under test. For example, the rejection of the null hypothesis of homoskedasticity may be due to a misspecified functional form for the conditional mean. An example is given in section [3.7.6](#). Second, with a very large sample, even trivial deviations from the null hypothesis of correct specification will cause the test to reject the null hypothesis. For example, if a previously omitted regressor has a very small coefficient, say, 0.000001, then even with an infinitely large sample the estimate will be sufficiently precise that we will always reject the null of zero coefficient.

### **3.7.1 Test of omitted variables**

In microeconomics, the most common approach to deciding on the adequacy of a model is a Wald-test approach that fits a richer model and determines whether the data support the need for a richer model. For example, we may add additional regressors to the model and test whether they have a zero coefficient.

The additional regressor may be a variable not already included, a transformation of a variable already included such as a quadratic in age, or a quadratic with interaction terms in age and education. If groups of regressors are included, such as a set of region dummies, `test` can be used after `regress` to perform a joint test of statistical significance.

In some branches of biostatistics, it is common to include only regressors with  $p < 0.05$ . In microeconomics, it is common instead to additionally include regressors that are statistically insignificant at, say, level 0.05 if economic theory or conventional practice includes the variable as a control. This reduces the likelihood of inconsistent parameter estimation due to omitted-variables bias at the expense of reduced precision in estimation.

### **3.7.2 Test of the functional form of the conditional mean**

The linear regression model specifies that the conditional mean of the dependent variable (whether measured in levels or in logs) equals  $\mathbf{x}'\beta$ . A standard test that this is the correct specification is the RESET variable augmentation test.

The `estat ovtest` postestimation command provides a RESET test that regresses  $y$  on  $\mathbf{x}$  and  $\hat{y}^2$ ,  $\hat{y}^3$ , and  $\hat{y}^4$  and jointly tests that the coefficients of  $\hat{y}^2$ ,  $\hat{y}^3$ , and  $\hat{y}^4$  are 0. We have

```
. * Variable augmentation test of conditional mean using estat ovtest
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. estat ovtest
Ramsey RESET test for omitted variables
Omitted: Powers of fitted values of ltotexp
H0: Model has no omitted variables
F(3, 2944) = 9.04
Prob > F = 0.0000
```

The model is strongly rejected because  $p = 0.000$ .

An alternative, simpler test is provided by the `linktest` command. This regresses  $y$  on  $\hat{y}$  and  $\hat{y}^2$ , where now the original model regressors  $\mathbf{x}$  are omitted, and it tests whether the coefficient of  $\hat{y}^2$  is 0. We have

```
. * Link test of functional form of conditional mean
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. linktest
```

Source	SS	df	MS	Number of obs	=	2,955
Model	1301.41696	2	650.708481	F(2, 2952)	=	454.81
Residual	4223.47242	2,952	1.43071559	Prob > F	=	0.0000
				R-squared	=	0.2356
Total	5524.88938	2,954	1.87030785	Adj R-squared	=	0.2350
				Root MSE	=	1.1961

ltotexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
_hat	4.429216	.6779517	6.53	0.000	3.09991 5.758522
_hatsq	-.2084091	.0411515	-5.06	0.000	-.2890976 -.1277206
_cons	-14.01127	2.779936	-5.04	0.000	-19.46208 -8.56046

Again, the null hypothesis that the conditional mean is correctly specified is rejected. A likely reason is that so few regressors were included in the model for pedagogical reasons.

The two preceding commands had different formats. The first test used the `estat ovtest` command, where `estat` produces various statistics following estimation, and the particular statistics available vary with the previous estimation command. The second test used `linktest`, which is available for a wider range of models.

### 3.7.3 Test of levels versus logs

A common specification-testing approach is to fit a richer model that tests the current model as a special case and performs a Wald test of the parameter restrictions that lead to the simpler model. The preceding omitted-variable test is an example.

Here we consider a test specific to the current example. We want to decide whether a regression model for medical expenditures is better in logs than in levels. There is no obvious way to compare the two models because they have different dependent variables. However, the Box–Cox transform leads to a richer model that includes the linear and log-linear models as special cases. Specifically, we fit the model with the transformed dependent variable

$$g(y_i, \theta) \equiv \frac{y_i^\theta - 1}{\theta} = \mathbf{x}'_i \boldsymbol{\beta} + u_i$$

where  $\theta$  and  $\boldsymbol{\beta}$  are estimated under the assumption that  $u_i \sim N(0, \sigma^2)$ . Three leading cases are 1)  $g(y, \theta) = y - 1$  if  $\theta = 1$ ; 2)  $g(y, \theta) = \ln y$  if  $\theta = 0$ ; and 3)  $g(y, \theta) = 1 - 1/y$  if  $\theta = -1$ . The log-linear model is supported if  $\hat{\theta}$  is close to 0, and the linear model is supported if  $\hat{\theta} = 1$ .

The Box–Cox transformation introduces a nonlinearity and an additional unknown parameter  $\theta$  into the model. This moves the modeling exercise into the domain of nonlinear models. The model is straightforward to fit, however, because Stata provides the `boxcox` command to fit the model. We obtain

```
. * Boxcox model with lhs variable transformed
. boxcox totexp suppins phyylim actlim totchr age female income if totexp>0, nolog
Fitting comparison model
Fitting full model
```

					Number of obs = 2,955
					LR chi2(7) = 773.02
					Prob > chi2 = 0.000
Log likelihood = -28518.267					
totexp	Coefficient	Std. err.	z	P> z	[95% conf. interval]
/theta	.0758956	.0096386	7.87	0.000	.0570042 .0947869

#### Estimates of scale-variant parameters

	Coefficient
Notrans	
suppins	.4459618
phyylim	.577317
actlim	.6905939
totchr	.6754338
age	.0051321
female	-.1767976
income	.0044039
_cons	8.930566
/sigma	2.189679

Test H0:	Restricted log likelihood	LR statistic chi2	Prob > chi2
theta = -1	-37454.643	17872.75	0.000
theta = 0	-28550.353	64.17	0.000
theta = 1	-31762.809	6489.08	0.000

The null hypothesis of  $\theta = 0$  is strongly rejected, so the log-linear model is rejected. However, the Box–Cox model with general  $\theta$  is difficult to interpret and use, and the estimate of  $\hat{\theta} = 0.0759$  gives much greater support for a log-linear model ( $\theta = 0$ ) than the linear model ( $\theta = 1$ ). Thus, we prefer to use the log-linear model.

### 3.7.4 Heteroskedasticity test

One consequence of heteroskedasticity is that default OLS standard errors are incorrect. This can be readily corrected and guarded against by routinely using heteroskedasticity-robust standard errors.

Nonetheless, there may be interest in formally testing whether heteroskedasticity is present. For example, the retransformation methods for the log-linear model used in section [4.2.3](#) assume homoskedastic errors. In section [6.3](#), we present diagnostic plots for heteroskedasticity. Here we instead present a formal test.

A quite general model of heteroskedasticity is

$$\text{Var}(y|\mathbf{x}) = h(\alpha_1 + \mathbf{z}'\boldsymbol{\alpha}_2)$$

where  $h(\cdot)$  is a positive monotonic function such as  $\exp(\cdot)$  and the variables in  $\mathbf{z}$  are functions of the variables in  $\mathbf{x}$ . Tests for heteroskedasticity are tests of

$$H_0: \boldsymbol{\alpha}_2 = \mathbf{0}$$

and can be shown to be independent of the choice of function  $h(\cdot)$ . We reject  $H_0$  at the  $\alpha$  level if the test statistic exceeds the  $\alpha$  critical value of a chi-squared distribution with degrees of freedom equal to the number of components of  $\mathbf{z}$ . The test is performed by using the `estat hettest` postestimation command. The simplest version is the Breusch–Pagan Lagrange multiplier test, which is equal to  $N$  times the uncentered explained sum of squares from the regression of the squared residuals on an intercept and  $\mathbf{z}$ . We use the `iid` option to obtain a different version of the test that relaxes the default assumption that the errors are normally distributed.

Several choices of the components of  $\mathbf{z}$  are possible. By far, the best choice is to use variables that are a priori likely determinants of heteroskedasticity. For example, in regressing the level of earnings on several regressors, including years of schooling, one will likely find that those with many years of schooling have the greatest variability in earnings.

Such candidates rarely exist. Instead, standard choices are to use the OLS fitted value  $\hat{y}$ , the default for `estat hettest`, or to use all the regressors so  $\mathbf{z} = \mathbf{x}$ . White's test for heteroskedasticity is equivalent to letting  $\mathbf{z}$  equal unique terms in the products and cross products of the terms in  $\mathbf{x}$ .

We consider tests of heteroskedasticity with both  $\mathbf{z} = \hat{y}$  and  $\mathbf{z} = \mathbf{x}$ . Then we have

```
. * Heteroskedasticity tests using estat hettest and option iid
. qui regress ltotexp suppins phylim actlim totchr age female income
. estat hettest, iid
Breusch-Pagan/Cook-Weisberg test for heteroskedasticity
Assumption: i.i.d. error terms
Variable: Fitted values of ltotexp
H0: Constant variance
    chi2(1) =  32.87
Prob > chi2 = 0.0000
. estat hettest suppins phylim actlim totchr age female income, iid
Breusch-Pagan/Cook-Weisberg test for heteroskedasticity
Assumption: i.i.d. error terms
Variables: suppins phylim actlim totchr age female income
H0: Constant variance
    chi2(7) =  93.13
Prob > chi2 = 0.0000
```

Both versions of the test, with  $\mathbf{z} = \hat{y}$  and with  $\mathbf{z} = \mathbf{x}$ , have  $p = 0.0000$  and strongly reject homoskedasticity.

### 3.7.5 Omnibus test

An alternative to separate tests of misspecification is an omnibus test, which is a joint test of misspecification in several directions. A leading example is the information matrix (IM) test (see section [11.9](#)), which is a test for correct specification of a fully parametric model based on whether the IM equality holds. For linear regression with normal homoskedastic errors, the IM test can be shown to be a joint test of heteroskedasticity, skewness, and nonnormal kurtosis compared with the null hypothesis of homoskedasticity, symmetry, and kurtosis coefficient of 3; see [Hall \(1987\)](#).

The `estat imtest` postestimation command computes the joint IM test and also splits it into its three components. We obtain

```
. * Information matrix test
. qui regress ltotexp suppins phylim actlim totchr age female income
. estat imtest
Cameron & Trivedi's decomposition of IM-test
```

Source	chi2	df	p
Heteroskedasticity	139.90	31	0.0000
Skewness	35.11	7	0.0000
Kurtosis	11.96	1	0.0005
Total	186.97	39	0.0000

The overall joint IM test rejects the model assumption that  $y \sim N(\mathbf{x}'\beta, \sigma^2 \mathbf{I})$  because  $p = 0.0000$  in the `Total` row. The decomposition indicates that all three assumptions of homoskedasticity, symmetry, and normal kurtosis are rejected. Note, however, that the decomposition assumes correct specification of the conditional mean. If instead the mean is misspecified, then that could be the cause of rejection of the model by the IM test.

### 3.7.6 Tests have power in more than one direction

Tests can have power in more than one direction, so that if a test targeted to a particular type of model misspecification rejects a model, it is not necessarily the case that this particular type of model misspecification is the underlying problem. For example, a test of heteroskedasticity may reject homoskedasticity, even though the underlying cause of rejection is that the conditional mean is misspecified rather than that errors are heteroskedastic.

To illustrate this example, we use the following simulation exercise. The DGP is one with homoskedastic normal errors,

$$y_i = \exp(1 + 0.25 \times x_i + 4 \times x_i^2) + u_i \\ x_i \sim U(0, 1), \quad u_i \sim N(0, 1)$$

We instead fit a model with a misspecified conditional mean function:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + v$$

We consider a simulation with a sample size of 50. We generate the regressors and the dependent variable by using commands detailed in section [5.2](#). We obtain

```
. * Simulation to show tests have power in more than one direction
. clear all
. set obs 50
Number of observations (_N) was 0, now 50.
. set seed 10101
. generate x = runiform() // x ~ uniform(0,1)
. generate u = rnormal() // u ~ N(0,1)
. generate y = exp(1 + 0.25*x + 4*x^2) + u
. generate xsq = x^2
. regress y x xsq
```

Source	SS	df	MS	Number of obs	=	50
Model	19660.0057	2	9830.00286	F(2, 47)	=	182.17
Residual	2536.17254	47	53.9611179	Prob > F	=	0.0000
				R-squared	=	0.8857
Total	22196.1783	49	452.98323	Adj R-squared	=	0.8809
				Root MSE	=	7.3458
<hr/>						
y	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
x	-101.1834	14.73595	-6.87	0.000	-130.8283	-71.53845
xsq	186.1681	16.56718	11.24	0.000	152.8392	219.4969
_cons	11.48277	2.590577	4.43	0.000	6.271202	16.69434

The misspecified model seems to fit the data very well with highly statistically significant regressors and an  $R^2$  of 0.89.

Now consider a test for heteroskedasticity:

```
. * Test for heteroskedasticity
. estat hettest
Breusch-Pagan/Cook-Weisberg test for heteroskedasticity
Assumption: Normal error terms
Variable: Fitted values of y
H0: Constant variance
    chi2(1) = 62.41
Prob > chi2 = 0.0000
```

This test strongly suggests that the errors are heteroskedastic because  $p = 0.0000$ , even though the DGP had homoskedastic errors.

The problem is that the regression function itself was misspecified. A RESET test yields

```
. * Test for misspecified conditional mean
. estat ovtest
Ramsey RESET test for omitted variables
Omitted: Powers of fitted values of y
H0: Model has no omitted variables
F(3, 44) = 824.55
Prob > F = 0.0000
```

This strongly rejects correct specification of the conditional mean because  $p = 0.0000$ .

Going the other way, could misspecification of other features of the model lead to rejection of the conditional mean, even though the conditional mean itself was correctly specified? This is an econometrically subtle question. The answer, in general, is yes. However, for the linear regression model, this is not the case essentially because consistency of the OLS estimator requires only that the conditional mean be correctly specified.

## 3.8 Sampling weights

The analysis to date has presumed simple random sampling, where sample observations have been drawn from the population with equal probability. In practice, however, many microeconomic studies use data from surveys that are not representative of the population. Instead, groups of key interest to policymakers that would have too few observations in a purely random sample are oversampled, with other groups then undersampled. Examples are individuals from racial minorities or those with low income or living in sparsely populated states.

As explained below, weights should be used for estimation of population means and for postregression prediction and computation of MES. However, in most cases, the regression itself can be fit without weights, as is the norm in microeconomics. If weighted analysis is desired, it can be done using standard commands with a weighting option, which is the approach of this section and the standard approach in microeconomics. Alternatively, one can use survey commands as detailed in section [6.9](#).

### 3.8.1 Weights

Sampling weights are provided by most survey datasets. These are called probability weights or `pweights` in Stata, though some others call them inverse-probability weights because they are inversely proportional to the probability of inclusion in the sample. A `pweight` of 1,400 in a survey of the U.S. population, for example, means that the observation is representative of 1,400 U.S. residents and that the probability of this observation being included in the sample is 1/1400.

Most estimation commands allow probability-weighted estimators that are obtained by adding `[pweight=weight]`, where *weight* is the name of the weighting variable.

To illustrate the use of sampling weights, we create an artificial weighting variable (sampling weights are available for the Medical Expenditure Panel Survey data but were not included in the data extract used

in this chapter). We manufacture weights that increase the weight given to those with more chronic problems. In practice, such weights might arise if the original sampling framework oversampled people with few chronic problems and undersampled people with many chronic problems. In this section, we analyze levels of expenditures, including expenditures of zero. Specifically,

```
. * Create artificial sampling weights
. qui use mus203mepsmedexp, clear
. generate swght = totchr^2 + 0.5
. summarize swght
```

Variable	Obs	Mean	Std. dev.	Min	Max
swght	3,064	5.285574	6.029423	.5	49.5

What matters in subsequent analysis is the relative values of the sampling weights rather than the absolute values. The sampling weight variable `swght` takes on values from 0.5 to 49.5, so weighted analysis will give some observations as much as  $49.5/0.5 = 99$  times the weight given to others.

Stata offers three other types of weights that for most analyses can be ignored. Analytical weights, called `aweights`, are used for the quite different purpose of compensating for different observations having different variances that are known up to scale; see section [6.3.4](#). For duplicated observations, `fweights` provide the number of duplicated observations. So-called importance weights, or `iweights`, are sometimes used in more advanced programming.

### 3.8.2 Weighted mean

If an estimate of a population mean is desired, then we should clearly weight. In this example, by oversampling those with few chronic problems, we will have oversampled people who on average have low medical expenditures, so that the unweighted sample mean will understate population mean medical expenditures.

Let  $w_i$  be the population weight for individual  $i$ . Then, by defining  $W = \sum_{i=1}^N w_i$  to be the sum of the weights, we see the weighted mean  $\bar{y}_W$

is

$$\bar{y}_W = \frac{1}{W} \sum_{i=1}^N w_i y_i$$

with variance estimator (assuming independent observations)

$\widehat{V}(\bar{y}_W) = \{1/W(W-1)\} \sum_{i=1}^N w_i (y_i - \bar{y}_W)^2$ . These formulas reduce to those for the unweighted mean if equal weights are used.

The weighted mean downweights oversampled observations because they will have a value of `pweights` (and hence  $w_i$ ) that is smaller than that for most observations. We have

Mean estimation		Number of obs = 3,064		
	Mean	Std. err.	[95% conf. interval]	
totexp	10670.83	428.5148	9830.62	11511.03

The weighted mean of \$10,671 is much larger than the unweighted mean of \$7,031 (see section [3.2.4](#)) because the unweighted mean does not adjust for the oversampling of individuals with few chronic problems.

### 3.8.3 Weighted regression

The weighted least-squares estimator for the regression of  $y_i$  on  $\mathbf{x}_i$  with the weights  $w_i$  is given by

$$\widehat{\boldsymbol{\beta}}_W = \left( \sum_{i=1}^N w_i \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \sum_{i=1}^N w_i \mathbf{x}_i y_i$$

The OLS estimator is the special case of equal weights with  $w_i = w_j$  for all  $i$  and  $j$ . The `regress` command when used with weights reports by default an estimator of the VCE that is a weighted version of the heteroskedasticity-robust version in (3.3), which assumes independent observations. So if errors are heteroskedastic, the `vce(robust)` option is redundant, while if observations are clustered, then the option `vce(cluster clustvar)` should be used.

For our data example, we obtain

```
. * Perform weighted regression
. regress totexp suppins phylim actlim totchr age female income [pweight=swght]
(sum of wgt is 16,195)

Linear regression
Number of obs      =      3,064
F(7, 3056)        =     14.08
Prob > F          =     0.0000
R-squared          =     0.0977
Root MSE           =    13824
```

totexp	Coefficient	Robust		t	P> t	[95% conf. interval]
		std. err.				
suppins	278.1578	825.6959	0.34	0.736	-1340.818	1897.133
phylim	2484.52	933.7116	2.66	0.008	653.7541	4315.286
actlim	4271.154	1024.686	4.17	0.000	2262.011	6280.296
totchr	1819.929	349.2234	5.21	0.000	1135.193	2504.666
age	-59.3125	68.01237	-0.87	0.383	-192.6671	74.04212
female	-2654.432	911.6422	-2.91	0.004	-4441.926	-866.9381
income	5.042348	16.6509	0.30	0.762	-27.60575	37.69045
_cons	7336.758	5263.377	1.39	0.163	-2983.359	17656.87

The estimated coefficients of all statistically significant variables aside from `female` are within 10% of those from unweighted regression (not given for brevity). Big differences between weighted and unweighted regression would indicate that  $E(u_i|\mathbf{x}_i) \neq 0$  because of model misspecification. Note that heteroskedastic-robust standard errors are reported by default.

Although the weighted estimator is easily obtained, for legitimate reasons many microeconomic analyses do not use weighted regression even where sampling weights are available. We provide a brief explanation of this conceptually difficult issue. For a more complete discussion, see [Cameron and Trivedi \(2005, 818–821\)](#) and [Solon, Haider, and Wooldridge \(2015\)](#).

Weighted regression should be used if a census parameter estimate is desired. For example, suppose we want to obtain an estimate for the U.S. population of the average change in earnings associated with one more year of schooling. Then, if disadvantaged minorities are oversampled, we most likely will underestimate the earnings increase because disadvantaged minorities are likely to have earnings that are lower than average for their given level of schooling. A second example is when aggregate state-level data are used in a natural experiment setting, where the goal is to measure the effect of an exogenous policy change that affects some states and not other states. Intuitively, the impact on more populous states should be given more weight. Note that these estimates are being given a correlative rather than a causal interpretation.

Weighted regression is not needed if we make the stronger assumptions that the DGP is the specified model  $y_i = \mathbf{x}'_i \boldsymbol{\beta} + u_i$  and sufficient controls are assumed to be added so that the error  $E(u_i|\mathbf{x}_i) = 0$ . This approach, called a model approach, is the approach usually taken in microeconometric studies that emphasize a causal interpretation of regression. Under the assumption that  $E(u_i|\mathbf{x}_i) = 0$ , the weighted least-squares estimator will be consistent for  $\boldsymbol{\beta}$  for any choice of weights including equal weights, and if  $u_i$  is homoskedastic, the most efficient estimator is the OLS estimator, which uses equal weights. For the assumption that  $E(u_i|\mathbf{x}_i) = 0$  to be reasonable, the determinants of the sampling frame should be included in the controls  $\mathbf{x}$  and should not be directly determined by the dependent variable  $y$ .

These points carry over directly to nonlinear regression models. In most cases, microeconometric analyses take on a model approach. In that case, unweighted estimation is appropriate, with any weighting based on efficiency grounds.

Leading cases where weights are used are the following. First, if a census-parameter approach is being taken, then it is necessary to weight. Second, for panel data where some individuals leave the panel over time, introducing possible selection bias, then we may weight by the inverse of the probability of leaving the panel; see section 19.11. Third, if individuals self-select into a treatment program, then one way to estimate the effect of treatment on an outcome, while controlling for this self-selection, is to

weight by the inverse of the predicted probability of self-selection into the program; see section 24.6.2.

### 3.8.4 Weighted prediction and MEs

After regression, unweighted prediction will provide an estimate of the sample-average value of the dependent variable. We may instead want to estimate the population-mean value of the dependent variable. Then sampling weights should be used in forming an average prediction.

This point is particularly easy to see for OLS regression. Because  $1/N \sum_i (y_i - \hat{y}_i) = 0$ , because in-sample residuals sum to zero if an intercept is included, the average prediction  $1/N \sum_i \hat{y}_i$  equals the sample mean  $\bar{y}$ . But given an unrepresentative sample, the unweighted sample mean  $\bar{y}$  may be a poor estimate of the population mean. Instead, we should use the weighted average prediction  $1/N \sum_i w_i \hat{y}_i$ , even if  $\hat{y}_i$  is obtained by using unweighted regression.

For this to be useful, however, the prediction should be based on a model that includes as regressors variables that control for the unrepresentative sampling.

For our example, we obtain the weighted prediction by typing

```
. * Weighted prediction
. qui predict yhatwols
. mean yhatwols [pweight=swght], noheader
```

	Mean	Std. err.	[95% conf. interval]
yhatwols	10670.83	138.0828	10400.08 10941.57

```
. mean yhatwols, noheader // Unweighted prediction
```

	Mean	Std. err.	[95% conf. interval]
yhatwols	7135.206	78.57376	6981.144 7289.269

The population mean for medical expenditures is predicted to be \$10,671 using weighted prediction, whereas the unweighted prediction gives a much

lower value of \$7,135.

Weights similarly should be used in computing average MES. For the linear model, the standard ME  $\partial E(y_i | \mathbf{x}_i) / \partial x_{ij}$  equals  $\beta_j$  for all observations, so weighting will make no difference in computing the ME. Weighting will make a difference for averages of other MES, such as elasticities, and for MES in nonlinear models.

## 3.9 OLS using Mata

Stata offers two different ways to perform computations using matrices: Stata `matrix` commands and Mata functions (which are discussed, respectively, in appendixes [A](#) and [B](#)).

Mata is much richer because it is a matrix programming language. We illustrate the use of Mata by using the same OLS regression as that in section [3.5.2](#).

The program is written for the dependent variable provided in the local macro `y` and the regressors in the local macro `xlist`. We begin by reading in the data and defining the local macros.

```
. * OLS with White robust standard errors using Mata
. qui use mus203mepsmedexp, clear
. keep if totexp > 0 // Analysis for positive medical expenditures only
(109 observations deleted)
. generate cons = 1
. local y ltotexp
. local xlist suppins phylim actlim totchr age female income cons
```

We then move into Mata. The `st_view()` Mata function is used to transfer the Stata data variables to Mata matrices `y` and `x`, with `tokens("")` added to convert '`xlist`' to a comma-separated list with each entry in double quotes, necessary for `st_view()`.

The key part of the program forms  $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$  and  $\hat{V}(\hat{\beta}) = (N/N - K) (\mathbf{X}'\mathbf{X})^{-1}(\sum_i \hat{u}_i^2 \mathbf{x}_i \mathbf{x}_i') (\mathbf{X}'\mathbf{X})^{-1}$ . The cross-product function `cross(x, x)` is used to form  $\mathbf{X}'\mathbf{X}$  because this handles missing values and is more efficient than the more obvious `x' * x`. The matrix inverse is formed by using `cholinv()` because this is the fastest method in the special case that the matrix is symmetric positive definite. We calculate the  $K \times K$  matrix  $\sum_i \hat{u}_i^2 \mathbf{x}_i \mathbf{x}_i'$  as  $\sum_i (\hat{u}_i \mathbf{x}_i')' (\hat{u}_i \mathbf{x}_i') = \mathbf{A}'\mathbf{A}$ , where the  $N \times K$  matrix `A` has an  $i$ th row equal to  $\hat{u}_i \mathbf{x}_i'$ . Now  $\hat{u}_i \mathbf{x}_i'$  equals the  $i$ th row of the  $N \times 1$  residual vector  $\hat{\mathbf{u}}$  times the  $i$ th row of the  $N \times K$  regressor matrix `X`, so `A` can be computed by element-by-element multiplication of `u` by `X`, or

$(e : *X)$ , where  $e$  is  $\hat{u}$ . Alternatively,  $\sum_i \hat{u}_i^2 \mathbf{x}_i \mathbf{x}'_i = \mathbf{X}' \mathbf{D} \mathbf{X}$ , where  $\mathbf{D}$  is an  $N \times N$  diagonal matrix with entries  $\hat{u}_i^2$ , but the matrix  $\mathbf{D}$  becomes exceptionally large, unnecessarily so, for a large  $N$ .

The Mata program concludes by using `st_matrix()` to pass the estimated  $\hat{\beta}$  and  $\hat{V}(\hat{\beta})$  back to Stata.

```
. mata
----- mata (type end to exit) -----
: // Create y vector and X matrix from Stata dataset
: st_view(y=., ., "`y'")           // y is nx1
: st_view(X=., ., tokens("`xlist'")) // X is nxk
: XXinv = cholinv(cross(X,X))      // XXinv is inverse of X'X
: b = XXinv*cross(X,y)            // b = [(X'X)^-1]*X'y
: e = y - X*b
: n = rows(X)
: k = cols(X)
: s2 = (e'e)/(n-k)
: vdef = s2*XXinv                // Default VCE not used here
: vwhite = XXinv*((e:*X)'(e:*X)*n/(n-k))*XXinv // Robust VCE
: st_matrix("b",b')               // Pass results from Mata to Stata
: st_matrix("V",vwhite)           // Pass results from Mata to Stata
: end
-----
```

Once back in Stata, we use `ereturn` to display the results in a format similar to that for official commands, first assigning name stripes to the columns and rows of `b` and `V`.

```

. * Use Stata ereturn display to present nicely formatted results
. matrix colnames b = `xlist'
. matrix colnames V = `xlist'
. matrix rownames V = `xlist'
. ereturn post b V
. ereturn display

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
suppins	.2556428	.0465982	5.49	0.000	.1643119 .3469736
phylim	.3020598	.057705	5.23	0.000	.18896 .4151595
actlim	.3560054	.0634066	5.61	0.000	.2317308 .48028
totchr	.3758201	.0187185	20.08	0.000	.3391326 .4125077
age	.0038016	.0037028	1.03	0.305	-.0034558 .011059
female	-.0843275	.045654	-1.85	0.065	-.1738076 .0051526
income	.0025498	.0010468	2.44	0.015	.0004981 .0046015
cons	6.703737	.2825751	23.72	0.000	6.1499 7.257575

The results are exactly the same as those given in section [3.5.2](#), when we used `regress` with the `vce(robust)` option.

## 3.10 Additional resources

The key Stata references are [U] *Stata User’s Guide* and [R] **regress**, [R] **regress postestimation**, [R] **estimates**, [R] **predict**, and [R] **test**. The material in this chapter appears in many econometrics texts, such as [Greene \(2018\)](#) and [Hansen \(2022a\)](#); see also [Hansen \(2022b\)](#) for univariate statistics.

Stata version 15 introduced a suite of commands, including `putdocx` and `dyndoc`, for exporting estimation results, summary statistics, and graphs to produce formatted reports in Word, Excel, PDF, and HTML. Furthermore, this can be done in a dynamic manner, so that any change in Stata output is immediately incorporated in the document. For details, see [RPT] *Stata Reporting Reference Manual*. For formatted output, Stata 17 introduced the `collect` command and a more flexible version of the `table` command.

## 3.11 Exercises

1. Fit the model in section 3.5 using only the first 100 observations. Compute standard errors in three ways: default, heteroskedastic, and cluster-robust where clustering is on the number of chronic problems. Use `estimates` to produce a table with three sets of coefficients and standard errors, and comment on any appreciable differences in the standard errors. Construct a similar table for three alternative sets of heteroskedasticity-robust standard errors, obtained by using the `vce(robust)`, `vce(hc2)`, and `vce(hc3)` options, and comment on any differences between the different estimates of the standard errors.
2. Fit the model in section 3.5 with robust standard errors reported. Test at 5% the joint significance of the demographic variables `age`, `female`, and `income`. Test the hypothesis that being male (rather than female) has the same impact on medical expenditures as aging 10 years. Fit the model under the constraint that  $\beta_{\text{phylim}} = \beta_{\text{actlim}}$  by first typing `constraint 1 phylim = actlim` and then by using `cnsreg` with the `constraints(1)` option.
3. Fit the model in section 3.6, and implement the RESET test manually by regressing  $y$  on  $x$  and  $\hat{y}^2$ ,  $\hat{y}^3$ , and  $\hat{y}^4$  and jointly testing that the coefficients of  $\hat{y}^2$ ,  $\hat{y}^3$ , and  $\hat{y}^4$  are 0. To get the same results as `estat ovtest`, do you need to use default or robust estimates of the VCE in this regression? Comment. Similarly, implement `linktest` by regressing  $y$  on  $\hat{y}$  and  $\hat{y}^2$  and testing that the coefficient of  $\hat{y}^2$  is 0. To get the same results as `linktest`, do you need to use default or robust estimates of the VCE in this regression? Comment.
4. Fit the model in section 3.6, and perform the standard Lagrange multiplier test for heteroskedasticity by using `estat hettest` with  $\mathbf{z} = \mathbf{x}$ . Then, implement the test manually as 0.5 times the explained sum of squares from the regression of  $y_i^*$  on an intercept and  $\mathbf{z}_i$ , where  $y_i^* = \{\hat{u}_i^2/(1/N) \sum_j \hat{u}_j^2\} - 1$  and  $\hat{u}_i$  is the residual from the original OLS regression. Next, use `estat hettest` with the `iid` option, and show that this test is obtained as  $N \times R^2$ , where  $R^2$  is obtained from the regression of  $\hat{u}_i^2$  on an intercept and  $\mathbf{z}_i$ .
5. Using the DGP of section 3.7.6, generate a sample of size 100. Regress  $y$  on  $x$  and an intercept. Apply the RESET omitted variable test. Does

the test indicate functional form misspecification? Next, use the `estat imtest` postestimation command to generate IM omnibus diagnostic test statistics. If you had applied only this test, what changes to your model specification would you make? Would you make additional tests before changing the model specification?

6. One type of data transformation that is sometimes used in linear regression, though not much in econometrics, is variable standardization. To standardize a variable you subtract the sample mean of that variable from each observation and divide the result by the sample standard deviation of that variable. The resulting variable with mean zero and standard deviation one is independent of the measurement scale of original data. A standardized regression is a regression in which the dependent and regressor variables are all standardized before running the regression. The standardized regression coefficients are interpreted as measuring the effect of one standard deviation change in the regressor and hence are in a sense comparable across variables. Note that when a regressor is discrete, interpreting its impact in units of standard deviation is not natural. Reestimate the regression equation of section [3.5.2](#) after standardizing all variables. Are the insignificant coefficients the same as those in the previous regression? Which variable has the largest coefficient? Does that mean it is the most important variable in the regression?



# **Chapter 4**

## **Linear regression extensions**

## 4.1 Introduction

In this chapter, we consider several extensions of linear regression that are widely used in applied work. These extensions are consequences of the reality that an econometric investigation rarely concludes with the straightforward reporting of a linear regression. More often, the regression is a tool used for a deeper analysis of the data, for testing economic hypotheses, for making predictions (in-sample or out-of-sample), and so forth. These topics are considered in this chapter.

The plausibility of an empirical application depends partially on the robustness of the fitted model. Hence, various tests considered in sections [3.6](#) and [3.7](#) are a natural preliminary to the applications considered in this chapter because they are designed to reveal weaknesses in the model specification. Ideally, one would fix the detected deficiencies of a model before proceeding with the applications considered in this chapter.

All the applications considered are based on the conditional mean function, rather than on the higher conditional moments. This raises the question whether the misspecification of other features of the model, including higher moments, may lead to inconsistent estimation of the conditional mean, even if the model for the conditional mean is correctly specified. This is an econometrically subtle question. The answer, in general, is yes. However, for the linear regression model, this is not the case essentially because consistency of the OLS estimator requires only that the conditional mean function be correctly specified.

Four main applications of linear regression are considered. These are prediction (in-sample and out-of-sample), estimation of various marginal effects (MES), regression decomposition, and estimation of treatment effects based on difference in differences (DID).

## 4.2 In-sample prediction

For the linear regression model, the estimator of the conditional mean of  $y$  given  $\mathbf{x} = \mathbf{x}_p$ ,  $E(y|\mathbf{x}_p) = \mathbf{x}'_p \boldsymbol{\beta}$ , is the conditional predictor  $\hat{y} = \mathbf{x}'_p \hat{\boldsymbol{\beta}}$ . We begin with prediction from a linear model for medical expenditures, because this is more straightforward, before turning to the log-linear model.

We present prediction for each observation in the estimation sample in this section and out-of-sample prediction in the subsequent section.

Further details on prediction are given in sections [13.5–13.7](#), where many methods are presented. Weighted average prediction to obtain a population-average prediction is presented in section [3.8](#).

### 4.2.1 The predict command

Predictions are obtained from the `predict` command after `regress`. The syntax for the `predict` command is

```
predict [ type ] newvar [ if ] [ in ] [ , statistic ]
```

The user always provides a name for the created variable, *newvar*.

The default is to predict for all observations in the sample, aside from any observations with missing regressor values. The qualifier `if e(sample)` ensures that predictions are made only for those observations used to obtain the estimates. For example, if the estimation sample was restricted to women, then the qualifier `if e(sample)` ensures prediction is only for women. Similarly, if data are available for all regressors but are missing for some observations of the dependent variable, then this qualifier ensures prediction is only for observations with nonmissing values of the dependent variable.

The available statistics vary with the estimation command preceding the `predict` command. The `predict` command can provide 16 different statistics following `regress`.

The default statistic `xb` is the prediction  $\hat{y}_i = \mathbf{x}'_i \hat{\beta}$ . Statistic `stdp` gives the standard error of  $\hat{y}_i$  as a prediction of  $E(y_i | \mathbf{x}_i)$ , and `stdf` gives the standard error of  $\hat{y}_i$  as a forecast of  $y_i$ .

The statistic `residuals` (or equivalently, `score`) yields  $y_i - \hat{y}_i$ , and `stdr` gives the standard error of the residual, while `rstandard` yields standardized residuals and `rstudent` yields studentized residuals. The statistics `cooksdi`, `leverage`, `dfbeta()`, `covratio`, `dfits`, and `welsch` are measures of observation leverage and influence.

The statistic `pr(a,b)` yields the probability that  $y_i$  lies in the range  $(a, b)$ , while `e(a,b)` yields the expected value of  $y_i$  if  $y_i$  is restricted to lie in the range  $(a, b)$ . The statistic `ystar(a,b)` yields the expected value of  $y_i^*$ , where  $y_i^* = a$  if  $y_i \leq a$ ,  $y_i^* = y_i$  if  $a < y_i < b$ , and  $y_i^* = b$  if  $y_i \geq b$ .

#### 4.2.2 In-sample prediction

We use the same data as that used in chapter 3 and begin with the linear regression model in levels rather than logs. Because we will compare results from the linear model with those from the model in logs, we restrict analysis to only those observations with positive medical expenditures.

OLS regression yields

```

. * OLS in levels for positive medical expenditures
. qui use mus203mepsmedexp
. keep if totexp > 0
(109 observations deleted)
. regress totexp suppins phylim actlim totchr age female income, vce(robust)

Linear regression
Number of obs      =      2,955
F(7, 2947)         =      40.58
Prob > F          =     0.0000
R-squared           =     0.1163
Root MSE            =    11285

```

totexp	Coefficient	Robust	t	P> t	[95% conf. interval]	
		std. err.				
suppins	724.8632	427.3045	1.70	0.090	-112.9824	1562.709
phylim	2389.019	544.3493	4.39	0.000	1321.675	3456.362
actlim	3900.491	705.2244	5.53	0.000	2517.708	5283.273
totchr	1844.377	186.8938	9.87	0.000	1477.921	2210.832
age	-85.36264	37.81868	-2.26	0.024	-159.5163	-11.20892
female	-1383.29	432.4759	-3.20	0.001	-2231.275	-535.3044
income	6.46894	8.570658	0.75	0.450	-10.33614	23.27402
_cons	8358.954	2847.802	2.94	0.003	2775.07	13942.84

We then predict the level of medical expenditures. Because the dependent variable is always available, the qualifier `if e(sample)` is redundant. We obtain

```

. * In-sample prediction following OLS in levels
. predict yhatlevels
(option xb assumed; fitted values)
. summarize totexp yhatlevels

```

Variable	Obs	Mean	Std. dev.	Min	Max
totexp	2,955	7290.235	11990.84	3	125610
yhatlevels	2,955	7290.235	4089.624	-236.3781	22559

The summary statistics show that on average the predicted value `yhatlevels` equals the dependent variable. This suggests that the predictor does a good job. But this is misleading because this is always the case for in-sample prediction after OLS regression in a model with an intercept, because then residuals sum to zero, implying  $\sum y_i = \sum \hat{y}_i$ . The standard deviation of `yhatlevels` is \$4,090, so there is some variation in the predicted values, though the variation is much less than that in the actual values.

A more discriminating comparison is, for example, comparison of the median predicted and median actual values. We have

- . \* Compare median prediction and sample median actual value after OLS
- . tabstat totexp yhatlevels, stat(count p50) col(stat)

Variable	N	p50
totexp	2955	3334
yhatlevels	2955	6464.692

There is considerable difference between the two. This is most likely due to the right skewness of the original data; the predictions are also right skewed but do not fully capture the right skewness.

#### 4.2.3 Prediction in logs: The retransformation problem

Transforming the dependent variable by taking the natural logarithm complicates prediction. We can directly predict  $E(\ln y|\mathbf{x})$ , but we are instead interested in  $E(y|\mathbf{x})$  because we want to predict the level of medical expenditures rather than the natural logarithm. The obvious procedure of predicting  $\ln y$  and taking the exponential is wrong because  $\exp\{E(\ln y)\} \neq E(y)$ , just as, for example,  $\sqrt{E(y^2)} \neq E(y)$ .

The log-linear model  $\ln y = \mathbf{x}'\boldsymbol{\beta} + u$  implies that  $y = \exp(\mathbf{x}'\boldsymbol{\beta}) \exp(u)$ . It follows that

$$E(y_i|\mathbf{x}_i) = \exp(\mathbf{x}'_i\boldsymbol{\beta})E\{\exp(u_i)\}$$

The simplest prediction is  $\exp(\mathbf{x}'_i\hat{\boldsymbol{\beta}})$ , but this leads to retransformation bias because it ignores the multiple  $E\{\exp(u_i)\}$ . If it is assumed that  $u_i \sim N(0, \sigma^2)$ , then it can be shown that  $E\{\exp(u_i)\} = \exp(0.5\sigma^2)$ , which can be estimated by  $\exp(0.5\hat{\sigma}^2)$ , where  $\hat{\sigma}^2$  is an unbiased estimator of the log-linear regression model error. A weaker assumption is to assume that  $u_i$  is independent and identically distributed, in which case we can consistently estimate  $E\{\exp(u_i)\}$  by the sample average  $N^{-1} \sum_{j=1}^N \exp(\hat{u}_j)$ ; see [Duan \(1983\)](#).

Applying these methods to the medical expenditure data yields

```
. * In-sample prediction in levels from a logarithmic model
. qui regress ltotexp suppins phylim actlim totchr age female income, vce(robust)
. qui predict lyhat
. generate yhatwrong = exp(lyhat)
. generate yhatnormal = exp(lyhat)*exp(0.5*e(rmse)^2)
. qui predict uhat, residual
. generate expuhat = exp(uhat)
. qui summarize expuhat
. generate yhatduan = r(mean)*exp(lyhat)

. summarize totexp yhatwrong yhatnormal yhatduan yhatlevels
```

Variable	Obs	Mean	Std. dev.	Min	Max
totexp	2,955	7290.235	11990.84	3	125610
yhatwrong	2,955	4004.453	3303.555	959.5991	37726.22
yhatnormal	2,955	8249.927	6805.945	1976.955	77723.13
yhatduan	2,955	8005.522	6604.318	1918.387	75420.57
yhatlevels	2,955	7290.235	4089.624	-236.3781	22559

Ignoring the retransformation bias leads to a very poor prediction because `yhatwrong` has a mean of \$4,004 compared with the sample mean of \$7,290. The two alternative methods yield much closer average values of \$8,250 and \$8,006. Furthermore, the predictions from log regression, compared with those in levels, have the desirable feature of always being positive and have greater variability. The standard deviation of `yhatnormal`, for example, is \$6,806 compared with \$4,090 from the levels model.

#### 4.2.4 Prediction exercise with a binary regressor

There are several different ways that predictions can be used to simulate the effects of a policy experiment. We consider the effect of a binary treatment, whether a person has supplementary insurance, on medical expenditure. This is an example of estimating an average treatment effect, the subject of chapters 24 and 25.

We base our predictions on estimates that assume supplementary insurance is exogenous. More realistically, supplementary insurance may be endogenous where, as we discuss in section [7.3.1](#), a variable is endogenous

if it is related to the error term. In that case, one should use other methods such as those presented in chapters 7 and 25. The simpler analysis here assumes that supplementary insurance is not related to the error term.

### Compare means of the dependent variable

An obvious approach is to compare the difference in sample means ( $\bar{y}_1 - \bar{y}_0$ ), where the subscript 1 denotes those with supplementary insurance and the subscript 0 denotes those without supplementary insurance.

```
. * Effect of suppins: (1) compare actual means with suppins==1 and suppins==0
. bysort suppins: summarize totexp
```

---

-> suppins = No

Variable	Obs	Mean	Std. dev.	Min	Max
totexp	1,207	6824.303	11425.94	9	104823

---

-> suppins = Yes

Variable	Obs	Mean	Std. dev.	Min	Max
totexp	1,748	7611.963	12358.83	3	125610

The average difference is \$788 (from 7612 – 6824). This measure has the major limitation that it does not control for individual characteristics.

### Compare means of predictions of the dependent variable

A measure that does control for individual characteristics is obtained by regression on `suppins` and additional control variables using all observations and by computing the difference in mean predictions ( $\hat{y}_1 - \hat{y}_0$ ), where  $\hat{y}_1$  and  $\hat{y}_0$  denote, respectively, the average prediction for those individuals with and without supplementary health insurance.

We implement this measure for the complete sample based on predictions obtained from the earlier OLS regressions in both levels and logs. We obtain

```
. * Effect of suppins: (2) compare predicted means by suppins value
. bysort suppins: summarize yhatlevels yhatduan
```

---

-> suppins = No

Variable	Obs	Mean	Std. dev.	Min	Max
yhatlevels	1,207	6824.303	4077.064	-236.3781	20131.43
yhatduan	1,207	6745.959	5365.255	1918.387	54981.73

---

-> suppins = Yes

Variable	Obs	Mean	Std. dev.	Min	Max
yhatlevels	1,748	7611.963	4068.397	502.9237	22559
yhatduan	1,748	8875.255	7212.993	2518.538	75420.57

Using predictions from OLS in levels, we see the average difference is again \$788 (from 7612 – 6824). This surprising equality of the difference in mean fitted values and the difference in sample means is a consequence of OLS regression in levels with an indicator variable and prediction using the estimation sample.

Using instead OLS in the log-linear model, with subsequent levels predictions that correct for retransformation bias using Duan's method, gives an average difference of \$2,129 (from 8875 – 6746), which differs from \$788.

#### Compare predictions at different values of the key regressor variable

A third measure is obtained by also regressing on `suppins` and additional control variables using all observations. Then 1) predict  $y$  with `suppins` set to 1 for all observations, and average these predictions; 2) predict  $y$  with `suppins` set to 0 for all observations, and average these predictions; and 3) compute the difference in these averages. This method is presented in more detail in section 24.4.

To implement this method, we need to make separate predictions for each individual with `suppins` set to 1 and with `suppins` set to 0. This requires changing the variable `suppins`, so we use `preserve` and `restore` (see section [2.5.2](#)) to eventually return `suppins` to its original sample values.

```

. * Effect of suppins: (3) predict with all set to suppins==0 versus all set to 1
. qui regress totexp suppins phylim actlim totchr age female income, vce(robust)
. preserve
. qui replace suppins = 1
. qui predict yhat1
. qui replace suppins = 0
. qui predict yhat0
. generate treateffect = yhat1 - yhat0
. summarize yhat1 yhat0 treateffect

```

Variable	Obs	Mean	Std. dev.	Min	Max
yhat1	2,955	7586.313	4071.367	488.4851	22559
yhat0	2,955	6861.45	4071.367	-236.3781	21834.14
treateffect	2,955	724.8632	.0002081	724.8623	724.8643

```

. restore

```

The estimate of \$725 is close to the preceding linear model estimate of \$788.

More precisely, the estimate is 724.8632, which equals the coefficient of `suppins` in the original OLS regression; see section [4.2.2](#). This equality is due to fitting a linear model with the regressor of interest changing its value by one unit.

This example includes control variables in the regression and then computes the average marginal effect (AME) of `suppins` using the finite-difference method; see section [3.5.9](#). It is simpler to use the `margins` command, which has the additional benefit of giving a standard error of the average marginal treatment effect.

```

. * Effect of suppins model: (3b) same as previous but easier and gives se
. qui regress totexp i.suppins phylim actlim totchr age female income, vce(robust)
. margins, dydx(suppins) nobylab

```

Average marginal effects		Number of obs = 2,955
Model VCE: Robust		
Expression: Linear prediction, predict()		
dy/dx wrt: 1.suppins		

	Delta-method				
	dy/dx	std. err.	t	P> t	[95% conf. interval]
1.suppins	724.8632	427.3045	1.70	0.090	-112.9824 1562.709

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

The `margins` command is presented in considerable length in sections [4.4](#) and [4.5](#).

**Compare predictions from different models for each value of the key regressor variable**

A refinement of the preceding method is to use two separate OLS regressions, one for each value of `suppins`, obtain predictions for all observations from each of these models, and compare the average predictions. We have

```
. * Effect of suppins model: (4) different fitted models for suppins =0 and =1
. qui regress totexp phylim actlim totchr age female income if suppins==0
. qui predict yhatmodel0
. qui regress totexp phylim actlim totchr age female income if suppins==1
. qui predict yhatmodel1
. generate treateffectra = yhatmodel1 - yhatmodel0
. summarize yhatmodel1 yhatmodel0 treateffectra
```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatmodel1	2,955	7605.52	4087.529	579.9417	22436.2
yhatmodel0	2,955	6935.549	4082.734	-428.2472	22108.39
treateffec^a	2,955	669.971	645.3849	-5050.99	2361.274

The average treatment effect is now \$670.

This method, called regression adjustment, is presented in section 24.6.1. The `teffects ra` command gives the same estimate of \$670 and additionally provides a standard error of the estimate.

#### 4.2.5 Forecast actual value versus prediction of conditional mean

Predictions given a regressor value  $\mathbf{x}_p$  can be used for two very distinct purposes.

First, we may wish to predict the conditional mean  $E(y|\mathbf{x}_p) = \mathbf{x}'_p \boldsymbol{\beta}$ . For example, on average, what level of health expenditure do we expect for individuals with characteristics  $\mathbf{x}_p$ ? Then we use the fitted value  $\hat{y}_p = \mathbf{x}'_p \hat{\boldsymbol{\beta}}$ , which is a noisy estimate because  $\hat{\boldsymbol{\beta}}$  is a noisy estimate of  $\boldsymbol{\beta}$ .

Second, we may wish to instead predict the actual value of  $y$  when  $\mathbf{x} = \mathbf{x}_p$ . This type of prediction is called a forecast. For example, what exact level of health expenditure do we expect for a single individual with characteristics  $\mathbf{x}_p$ . This is more challenging because then we wish to predict  $y_p = \mathbf{x}'_p \beta + u_p$ . Because  $u_p$  is pure noise our best estimate is its mean of zero, so we forecast  $y_p$  using  $\mathbf{x}'_p \hat{\beta} + 0$ . So we again use  $\hat{y}_p = \mathbf{x}'_p \hat{\beta}$ , but there is now much more noise because of the additional need to predict  $u_p$ . In fact, the variance of the forecast will be at least the variance of the error  $u_p$ , regardless of how precisely  $\beta$  is estimated.

The standard error of the prediction used as an estimate of the conditional mean can be obtained using the `stdp` option of the `predict` command. Obtaining the standard error of the prediction used as a forecast is more problematic because it requires an estimate of  $\text{Var}(u_p)$ . The `stdf` option of the `predict` command provides the standard error of a forecast only if default standard errors are used, because then given homoskedasticity, the standard deviation of  $u_p$  can be estimated using the standard error of the regression.

We therefore reestimate without the `vce(robust)` option and use `predict` to obtain

```
. * Compute standard errors of prediction and forecast with default VCE
. qui regress totexp suppins phylim actlim totchr age female income
. display "Estimated standard deviation of the error: s = " e(rmse)
Estimated standard deviation of the error: s = 11285.257
. predict yhatstdp, stdp
. predict yhatstdf, stdf
. summarize yhatlevels yhatstdp yhatstdf
```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatlevels	2,955	7290.235	4089.624	-236.3781	22559
yhatstdp	2,955	572.7	129.6575	393.5964	2813.983
yhatstdf	2,955	11300.52	10.50946	11292.12	11630.8

The first quantity, `yhatstdp`, views  $\mathbf{x}'_i \hat{\beta}$  as an estimate of the conditional mean  $\mathbf{x}'_i \beta$  and is quite precisely estimated; the prediction has average standard deviation of \$573, which is small compared with the average prediction of \$7,290.

The second quantity, `yhatstdf`, views  $\mathbf{x}_i' \hat{\boldsymbol{\beta}}$  as an estimate of the actual value  $y_i$  and is very imprecisely estimated with average standard deviation equal to \$11,301. This is because the error  $u_i$  here has large estimated standard deviation of  $s = 11285$  in the fitted OLS model. The standard deviation of the forecast is necessarily at least 11285. For further details and formulas, see section [4.3](#).

More generally, microeconometric models can predict poorly for a given individual, as evidenced by the typically low values of  $R^2$  obtained from regression on cross-sectional data. These same models may nonetheless predict the conditional mean well, and it is this latter quantity that is needed for policy analysis that focuses on average behavior.

## 4.3 Out-of-sample prediction

We now consider out-of-sample prediction. While we focus on the precision of predictions, note that out-of-sample predictions become more questionable as regressor values become further away from those used to fit the model. Accuracy of out-of-sample predictions is often a basis of a demanding test of model specification but often may not provide a clear indication of the deficient feature of a potentially flawed model.

### 4.3.1 Out-of-sample predictions

Out-of-sample predictions are useful for generating predictions per se but also for testing the fitted model against a subsample that was not used to fit the model. Some tests of model misspecification are based on out-of-sample prediction errors. The `predict` command can generate both in-sample and out-of-sample predictions.

Let  $\mathbf{X}_p$  denote the  $q \times K$  matrix for a set of  $q$  observations that were not used in estimation and held back to generate predictions outside the estimation sample. We suppose the dependent variable is generated by the same model as that fitted, so the  $q$  out-of-sample  $y$ 's are generated by  $\mathbf{y}_p = \mathbf{X}_p\boldsymbol{\beta} + \mathbf{u}_p$ . Hence, we predict the conditional mean and forecast the actual value using  $\hat{\mathbf{y}}_p = \mathbf{X}_p\hat{\boldsymbol{\beta}}$ . Then the vector of  $q$  prediction errors is  $\hat{\mathbf{u}}_p = \hat{\mathbf{y}}_p - \mathbf{y}_p$ . These prediction errors can be used to evaluate the fitted regression in a number of ways.

For the prediction exercise to be meaningful, the matrix  $\mathbf{X}_p$  should be “similar to” the one used in estimation; that is, its range of variation in the prediction sample should be similar to that of the estimation sample. Otherwise, the estimated sample estimates may be inappropriate for the prediction sample, and the divergence between regressors in the estimation and prediction samples will contribute significantly to the prediction error. Prediction-based tests may be used to test for sample homogeneity.

Predictions of the conditional mean have variance matrix

$$\text{Var} \left\{ \widehat{E}(\mathbf{y}_p | \mathbf{X}_p) \right\} = \text{Var} \left( \mathbf{X}_p \widehat{\boldsymbol{\beta}} \right) = \mathbf{X}_p \text{Var} \left( \widehat{\boldsymbol{\beta}} \right) \mathbf{X}'_p$$

This formula is used for calculating hypothesis tests and confidence intervals for the point-predicted conditional means. It can be used with default standard errors or with robust standard errors. Note that as the sample size gets larger,  $\text{Var}(\widehat{\boldsymbol{\beta}})$  gets smaller, so the precision of the prediction improves.

Forecasts of the actual value additionally need to predict the error term because  $\mathbf{y}_p = \mathbf{X}_p \boldsymbol{\beta} + \mathbf{u}_p$ . Our best estimate is  $\mathbf{u}_p = \mathbf{0}$ . Because the error is independent of the regressors, it follows that forecasts of the actual values have variance matrix

$$\text{Var} (\widehat{\mathbf{y}}_p | \mathbf{X}_p) = \text{Var} \left( \mathbf{X}_p \widehat{\boldsymbol{\beta}} + \mathbf{u}_p \right) = \mathbf{X}_p \text{Var} \left( \widehat{\boldsymbol{\beta}} \right) \mathbf{X}'_p + \text{Var}(\mathbf{u}_p)$$

This is more problematic to estimate because it depends on  $\text{Var}(\mathbf{u}_p)$ . Furthermore, note that even in large samples, there can be considerable forecast error because  $\text{Var}(\widehat{\mathbf{y}}_p) > \text{Var}(\mathbf{u}_p)$  always.

If the errors are independent and identically distributed (i.i.d.), then  $\text{Var}(\mathbf{u}_p) = \sigma^2 \mathbf{I}$  and  $\text{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2 (\mathbf{X}' \mathbf{X})^{-1}$ , so the variance of the forecast simplifies to  $\sigma^2 \{ \mathbf{I} + \mathbf{X}_p (\mathbf{X}' \mathbf{X})^{-1} \mathbf{X}'_p \}$ . Then, in the case of a single out-of-sample observation, a large-sample 95% confidence interval for a forecast of the actual value of the dependent variable when  $\mathbf{x} = \mathbf{x}_p$  is

$$\widehat{y}_p \pm 1.96 \sqrt{s^2 \{ 1 + \mathbf{x}'_p (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x}_p \}}$$

where  $s^2$  is the sample estimate of  $\sigma^2$ . The corresponding large-sample 95% confidence interval for the prediction of the conditional mean when  $\mathbf{x} = \mathbf{x}_p$  is

$$\hat{y}_p \pm 1.96 \sqrt{s^2 \mathbf{x}'_p (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x}_p}$$

The point prediction and the estimated variance  $s^2$  may both be sensitive to the presence of outliers generated by a mechanism different from the one that prevailed in the sample period.

The preceding considerations motivate the use of diagnostic procedures for testing homoskedasticity and normality of residuals mentioned in a previous subsection. The normal distribution is symmetric, and hence its third moment is 0. The standard normal distribution has fourth moment equal to 3. Hence, departures from normality are indicated by a skewness coefficient significantly different from 0 and a nonnormal kurtosis coefficient significantly greater than 3; for an example, see section [3.7.5](#).

### 4.3.2 Out-of-sample prediction example

The example given here is based on generated data with 100 observations. The model is fit to the subsample of the first 90 observations, and the remaining 10 are used for prediction. We show how out-of-sample prediction is implemented.

In the first part, the 100 observations are generated using the regression  $y = 1 + x + u$ . The correctly specified regression is fit to the first 90 observations, and 10 out-of-sample predictions are generated. For comparison, the regression is also fit to the full sample, and the corresponding 10 in-sample predictions are also generated. The prediction errors as well as fitted values are summarized. In this example, the data-generating process (DGP) satisfies the underlying assumptions, so the prediction errors reflect the estimation error and the intrinsic randomness due to the error term.

```

. * Out-of-sample OLS predictions using correctly specified model
. clear all
. qui set obs 100
. set seed 10101
. generate u = 3*rnormal()
. generate x = 10 + 4*rnormal()
. generate y = 1 + x + u
. qui regress y x if _n < 91
. qui predict yhatos if _n > 90 // Predict 10 out-of-sample observations
. qui regress y x
. qui predict yhatis if _n > 90 // Predict 10 in-sample observations
. qui generate ferroros = y - yhatos

. summarize y yhatis yhatos ferroros if _n > 90

```

Variable	Obs	Mean	Std. dev.	Min	Max
y	10	10.96446	2.697358	6.751842	16.415
yhatis	10	12.57903	3.894589	3.96609	19.10511
yhatos	10	12.85367	4.134129	3.710977	19.78113
ferroros	10	-1.889211	3.188283	-6.584855	4.737788

The 10 out-of-sample predictions have mean and standard deviation of (12.854, 4.134) compared with the corresponding 10 in-sample predictions using all 100 observations of (12.579, 3.895). The sample mean of the prediction error (`ferroros`) is expected to approach zero as the sample size gets larger, in the current example with correctly specified model.

We continue with data generated by the DGP  $y = 1 + x + z + u$  but generate predictions from estimation of a misspecified model that omits variable  $z$ , that is, assuming  $y = 1 + x + u$ . By construction, the omitted variable is uncorrelated with  $x$ , so the OLS regression coefficients should be unbiased, but the predictions will be biased, and the prediction errors will no longer have average close to zero in large samples.

```

. * Out-of-sample prediction under misspecification (omitted regressor)
. generate z = 10 + 4*rnorm()
. generate ynew = 1 + x + z + u
. qui regress ynew x if _n <91      // Regression with variable z omitted
. qui predict ynewhatos if _n >90 // Predict 10 out-of-sample observations
. qui regress ynew x
. qui predict ynewhatis if _n > 90 // Predict 10 in-sample observations
. qui generate ferrornewos = ynew - ynewhatos if _n > 90
. list ferrornewos in 91/100, clean
    ferrornewos
91.      4.19217
92.      1.800138
93.     -3.714031
94.      4.681175
95.     -5.369816
96.     -2.724915
97.     -7.702372
98.    -13.30714
99.     -8.065577
100.    -9.890109
. summarize ynew ynewhatis ynewhatos ferrornewos if _n > 90

```

Variable	Obs	Mean	Std. dev.	Min	Max
ynew	10	18.85934	5.791399	10.47607	28.13668
ynewhatis	10	22.35754	3.922385	13.68313	28.93019
ynewhatos	10	22.86938	4.232142	13.50993	29.96109
ferrornewos	10	-4.010048	6.066775	-13.30714	4.681175

The estimated mean prediction error is  $-4.010$ , indicating that predicted value is an underestimate of the realized value. The range of the prediction error is  $(-13.307, 4.681)$ , which is also greater relative to the correct specification case. Of course, this reflects the impact of  $z$ , and it would be smaller or larger depending on its importance relative to  $x$ .

Prediction errors are an obvious metric for measuring the performance of a regression model. However, because prediction errors are expected to be i.i.d. under correct specification of the DGP, they can be used for checking model specification. For example, given i.i.d. model errors, the chi-squared distributed statistic  $\hat{\mathbf{u}}_p'[\sigma^2\{\mathbf{I} + \mathbf{x}_p(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_p'\}^{-1}]\hat{\mathbf{u}}_p$  provides a basis for a formal test, although graphical displays are potentially also helpful in indicating weaknesses in the model specification.

The use of out-of-sample prediction for model selection, notably  $k$ -fold cross-validation, is detailed in section 28.2.

## 4.4 Predictive margins

Following linear regression, interest may lie in evaluating the conditional mean at different values of the regressors. For example, following regression that includes many control variables, we may wish to predict medical expenditure separately for men and for women, or we may wish to predict at several different ages. These are examples of predictive means (PMs), which will vary with the value assigned to the regressors.

More generally, quantities other than the conditional mean may be studied, such as conditional probabilities following logistic regression. [Lane and Nelder \(1982\)](#) introduced the term predictive margins to cover postestimation prediction of a quantity of interest at various values of the regressors.

### 4.4.1 Predictive margins

Following linear regression, a PM can be evaluated at a single point  $\mathbf{x} = \mathbf{x}^*$ , in which case  $\text{PM} = \mathbf{x}^*'\hat{\boldsymbol{\beta}}$ , or at different regressor values for different individuals and averaged. When evaluated at values  $\mathbf{x}_i = \mathbf{x}_i^*, i = 1, \dots, N$ , the PM is

$$\text{PM} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^*'\hat{\boldsymbol{\beta}}$$

In general, predictive margins of  $g(\mathbf{x}, \boldsymbol{\beta})$  are computed as  $(1/N) \sum_{i=1}^N g(\mathbf{x}_i^*, \hat{\boldsymbol{\beta}})$ .

There are many ways to specify the evaluation points  $\mathbf{x}_i^*$ . Leading cases are 1) compute PM at sample values  $\mathbf{x}_i, i = 1, \dots, N$ , and average; 2) compute at the sample mean  $\bar{\mathbf{x}}$ , in which case  $\text{PM} = \bar{\mathbf{x}}'\hat{\boldsymbol{\beta}}$ ; or 3) compute at a specified value  $\mathbf{x}^*$ , in which case  $\text{PM} = \mathbf{x}^*'\hat{\boldsymbol{\beta}}$ .

In practice, for models with multiple regressors, combinations of these methods are often used. Consider predictive margins for various specified values of  $\mathbf{z}$ , often a scalar variable but possibly more than one variable, where  $\mathbf{z}$  is a subcomponent of the regressors in the fitted model

$$\hat{y}_i = \mathbf{x}'_i \hat{\beta} = \mathbf{z}'_i \hat{\alpha} + \mathbf{w}'_i \hat{\gamma}$$

Let  $\mathbf{z}_k$  denote the  $k$ th-specified value of  $\mathbf{z}$ , in which case

$$(\hat{y}_i | \mathbf{z} = \mathbf{z}_k, \mathbf{w}_i = \mathbf{w}_i^*) = \mathbf{z}'_k \hat{\alpha} + \mathbf{w}'_i^* \hat{\gamma}$$

Then  $\mathbf{x}_i^{*'} = [\mathbf{z}'_k \ \mathbf{w}'_i^*]$ , and there are many ways to specify the evaluation points  $\mathbf{x}_i^*$  because there are many ways to specify values for the remaining variables  $\mathbf{w}_i^*$ .

First, we may set the remaining regressors at their sample values, so  $\mathbf{w}_i^* = \mathbf{w}_i$ . Second, we may set the remaining regressors at a common specified value  $\mathbf{w}^*$ , so  $\mathbf{w}_i^* = \mathbf{w}^*$ . Third, we may set the remaining regressors at their sample mean values, so  $\mathbf{w}_i^* = \bar{\mathbf{w}}$ . Finally, any combination of these is possible. For example, letting  $\mathbf{w}'_i \gamma = \mathbf{w}'_{1i} \gamma_1 + \mathbf{w}'_{2i} \gamma_2 + \mathbf{w}'_{3i} \gamma_3$ , we might evaluate at  $\mathbf{w}_i^{*'} = [\mathbf{w}'_{1i} \ \mathbf{w}'_{2i} \ \bar{\mathbf{w}}'_3]$ . Then some of the remaining regressors are set to sample values, some to specified values, and some to sample means.

In principle, predictive margins can be computed using one or more Stata commands, usually `generate` and `predict` commands. And standard errors of these prediction margins can be obtained using an appropriate bootstrap.

Much simpler is to directly use the powerful `margins` command. This command computes predictive margins and their associated standard errors,  $t$  statistics, and confidence intervals. It additionally can be used to compute MES, and the `marginsplot` command presents results graphically. The discussion is brief compared with the more than 100 pages of documentation given in [R] **margins**.

## 4.4.2 The margins command

The `margins` command after essentially any estimation command, including `regress`, has syntax

```
margins [ marginlist ] [ if ] [ in ] [ weight ] [ , response_options options ]
```

Here *marginlist* is a list of factor-indicator variables and any associated interactions defined using factor-variable operators. If *marginlist* is left blank, then the overall margin is reported.

The default *response\_option* for the `margins` command is to use the quantity that is the default for the postestimation command `predict`. In general, the *response\_option* can be specified to be one of the other quantities calculated by the postestimation command `predict`, or a function thereof. In general, however, not all the `predict` options are available for `margins`, and the only available `predict` option for `margins` following `regress` is the default `xb`. Other *response\_options* compute MES (option `dydx()`) and elasticities; these are presented in the subsequent section.

Key *options* are those that define the regressor values at which the margins are estimated. The `at()` option estimates margins at specified values of regressors. The `atmeans` option evaluates margins at the means of the regressors. For regressors not appearing in the `at()` option, when `atmeans` is not specified, the margins are estimated at sample values, and the predictive margin is computed by averaging over the estimation sample. Thus the sample average prediction is obtained if `atmeans` is not specified and some regressors are not included in the `at()` option, whereas the prediction at the sample mean is obtained if the `atmeans` option only is used.

Other options include `over(varlist)` to estimate margins at unique values of *varlist* and `subpop()` to estimate margins for subpopulations.

The output from the `margins` command includes standard errors and associated statistics for inference. The default standard errors are obtained by applying the delta method to the coefficient estimate standard errors computed from the preceding estimation command.

The default standard errors of margins estimates treat the regressors as fixed. This is fine if the value of every regressor is specified using the `at()` or `atmeans` option or if sample values are used and inference is on the in-sample predictive margin. But if a population predictive margin is desired and sample values of regressors are used in computing the predictive margin, then the `vce(unconditional)` option additionally controls for variation in the regressors due to sampling; see section [13.7.9](#).

If population-average margins are desired and population weights are provided, then one needs to compute weighted margins. If `weight` was used in the preceding estimation command, then weighted margins are automatically given. If the preceding estimation command did not use weights, as is commonly the case in microeconomics, then one needs to subsequently add `weight` to the `margins` command if weighted margins are desired.

The `noesample` option allows predictive margins to be computed for samples other than the estimation sample.

The `margins`, `contrast` command performs contrasts of margins. The `margins`, `pwcompare` command performs pairwise comparison of margins. The `marginsplot` command graphs the result of the preceding `margins` command. The `margins`, `dydx()` command computes MES.

#### 4.4.3 Predictive margins examples

The following examples consider only OLS estimation, with factor variables introduced to allow analysis in the presence of a quadratic in age and an interaction of gender and income. Then the linear model includes explanatory variables that enter nonlinearly. The examples consider only predicted conditional means and changes in predicted conditional means as regressors change. But these examples extend immediately to other estimators, such as logit, and to predictions of other quantities, such as predicted probabilities.

Consider the simple examples of using the `margins` command following the OLS regression command `regress y x i.z`. Recall that the default for

the `predict` command following the `regress` command yields the OLS prediction  $\hat{y}$ .

The `margins` command gives the sample average prediction  
 $\bar{\hat{y}} = (1/N) \sum_{i=1}^N \hat{y}_i$ .

The `margins z` command computes the sample average prediction  $\bar{\hat{y}}$  at each distinct value taken by the categorical variable  $z$ . In the simplest case where the factor variable `i.z` is a binary regressor taking values 0 and 1 the fitted model is  $\hat{y} = \hat{\beta}_1 + \hat{\beta}_2 x + \hat{\beta}_3 z$ . For  $z = 0$ , the predictive margin is  $(1/N) \sum_{i=1}^N (\hat{\beta}_1 + \hat{\beta}_2 x_i)$ , and for  $z = 1$ , the predictive margin is  $(1/N) \sum_{i=1}^N (\hat{\beta}_1 + \hat{\beta}_2 x_i + \hat{\beta}_3)$ .

For the regressor `x`, the `margins x` command will fail because variable `x` was not declared to be a categorical factor variable. Similar problems arise if we had used command `regress y c.x i.z`. Instead, we need to use the `at()` option to compute the predictive margin at various specified values of `x`. For example, the command `margins, at(x=(30(10)50))` computes  $(1/N) \sum_{i=1}^N (\hat{\beta}_1 + \hat{\beta}_2 x^* + \hat{\beta}_3 z_i)$  at, respectively,  $x^* = 30, 40, and }50.$

#### 4.4.4 Predictive margins for a categorical factor variable

We consider regression of `totexp` on various regressors, including a quadratic in `age` and interaction of `female` and `income`. The model is the same as that in section 4.2, except that now the dependent variable `totexp` is analyzed using a sample that includes the 109 observations with `totexp = 0`. For completeness, all regressors are defined using factor-variable operators, though for the following analysis, this is unnecessary for many of the regressors.

OLS regression yields the following estimated coefficients:

```

. * Factor variables used to define model with interactions and quadratic
. qui use mus203mepsmedexp, clear
. regress totexp i.suppins i.phylim i.actlim c.totchr c.age##c.age
>      i.female##c.income, vce(robust) noheader nofvlabel

```

totexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
1.suppins	854.581	413.6253	2.07	0.039	43.56894	1665.593
1.phylim	2432.789	536.4219	4.54	0.000	1381.004	3484.573
1.actlim	3779.804	690.7463	5.47	0.000	2425.429	5134.178
totchr	1921.016	180.5999	10.64	0.000	1566.906	2275.125
age	1342.275	763.421	1.76	0.079	-154.5957	2839.146
c.age#c.age	-9.452011	5.041687	-1.87	0.061	-19.33745	.433432
1.female	-1573.508	573.3188	-2.74	0.006	-2697.638	-449.3781
income	2.090179	11.27985	0.19	0.853	-20.02669	24.20705
female#c.income						
1	15.42391	15.81709	0.98	0.330	-15.58931	46.43713
_cons	-45432.3	28774.04	-1.58	0.114	-101850.7	10986.15

The stand-alone `margins` command gives the average prediction of `totexp` in the estimation sample.

```

. * Predictive margin using sample average
. margins
Predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()

```

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
_cons	7030.889	200.6571	35.04	0.000	6637.453	7424.326

For OLS regression with an intercept, the residuals sum to 0 so  $\hat{y} = \bar{y}$ , and the average predicted medical expenditure of \$7,031 equals the average of `totexp` in the estimation sample. More generally, for most estimation methods,  $\hat{y} \neq \bar{y}$ .

If we wish to use this as an estimate for the population, we should use the option `vce(unconditional)` to account for variability in the sample regressors. We have

. * Predictive margin using sample average with population-average inference					
. margins, vce(unconditional)					
Predictive margins	Number of obs = 3,064				
Expression: Linear prediction, predict()					
<hr/>					
	Unconditional				
	Margin	std. err.	t	P> t	[95% conf. interval]
_cons	7030.889	214.4439	32.79	0.000	6610.42 7451.358

The standard error has increased from 200.7 to 214.4, a 7% increase.

We next compute the estimation sample average prediction of `totexp` for the different values taken by the categorical variable `female`.

. * Predictive margins by categorical regressor gender using sample averages					
. margins female, nofvlabel					
Predictive margins	Number of obs = 3,064				
Model VCE: Robust					
Expression: Linear prediction, predict()					
<hr/>					
	Delta-method				
	Margin	std. err.	t	P> t	[95% conf. interval]
female					
0	7764.118	337.7036	22.99	0.000	7101.968 8426.267
1	6537.258	245.379	26.64	0.000	6056.133 7018.383

The average predicted medical expenditure for this sample of older people is \$7,764 for men and \$6,537 for women, so it is higher for men. This computation allows for variable `female` entering directly and through the interaction term with `income`. Because `female` enters interactively, the predicted gender difference of 7764 – 6537 differs from the raw data gender difference of 7452 – 6725.

The two 95% confidence intervals are nonoverlapping, which suggests, though does not guarantee, that the gender difference is statistically significant at level 0.05. A formal test is possible using the `margins`, `pwcompare` or `margins, contrast(ci)` command. This test, given in section [4.4.8](#), finds a statistically significant difference at level 0.05.

In the next example, the predictive margins by gender are computed at sample mean values of the remaining regressors, using the `atmeans` option.

```
. * Predictive margins by categorical regressor gender using sample means
. margins female, atmeans nofvlabel

Adjusted predictions                                         Number of obs = 3,064
Model VCE: Robust

Expression: Linear prediction, predict()
At: 0.suppins = .4187337 (mean)
    1.suppins = .5812663 (mean)
    0.phylim = .5744125 (mean)
    1.phylim = .4255875 (mean)
    0.actlim = .7163838 (mean)
    1.actlim = .2836162 (mean)
    totchr = 1.754243 (mean)
    age = 74.17167 (mean)
    0.female = .4203655 (mean)
    1.female = .5796345 (mean)
    income = 22.47472 (mean)
```

---

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
female	0	8147.88	391.5846	20.81	0.000	7380.084
	1	6921.02	313.0471	22.11	0.000	6307.216

---

Now the predicted medical expenditure is \$8,148 for men and \$6,921 for women, compared with, respectively, \$7,764 and \$6,537 in the preceding example, which used sample values of variables other than `female` and then averaged over the estimation sample. This difference between prediction at the average and an averaged prediction usually arises; a notable exception is OLS in a purely linear model that has no nonlinearities such as quadratic and interaction terms.

Predictive margins by gender can be computed at specified values of some of or all the remaining regressors, using the `at()` option. Here we obtain predictive margins by gender with values specified for all the remaining regressors.

```

. * Predictive margins by categorical regressor gender at specified values
. margins female, at(suppins=1 phylim=1 actlim=1 totchr=2 age=70 income=50)
> nofvlabel

Adjusted predictions                                         Number of obs = 3,064
Model VCE: Robust

Expression: Linear prediction, predict()
At: suppins = 1
    phylim = 1
    actlim = 1
    totchr = 2
    age    = 70
    income = 50

```

		Delta-method				
		Margin	std. err.	t	P> t	[95% conf. interval]
female	0	13225.83	793.4232	16.67	0.000	11670.13 14781.53
	1	12423.52	772.1657	16.09	0.000	10909.5 13937.53

In this example, evaluation was for a person with functional limitations, activity limitations, and two chronic conditions. As expected, this leads to above-average predicted medical expenditures. Again, expenditures are higher for men.

#### 4.4.5 Predictive margins for continuous variables

We now consider a continuous variable, `age`, where by “continuous” we mean a variable that is not specified to be a categorical factor variable in the original OLS regression.

Then the command `margins age` fails. Instead, we need to use the `at()` option to evaluate the predictive margins for various values of variable `age`.

The following example obtains predictive margins at ages 65, 75, and 85, with the remaining variables evaluated at their sample values.

```

. * Margins at 3 different ages (averaged over other regressors' sample values)
. margins, at(age=(65(10)85))
Predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
1._at: age = 65
2._at: age = 75
3._at: age = 85

```

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
_at						
1	7168.639	562.4323	12.75	0.000	6065.855	8271.423
2	7358.576	282.7122	26.03	0.000	6804.25	7912.901
3	5658.11	457.4907	12.37	0.000	4761.089	6555.131

For each specified value of `age`, the predictive margins are computed by evaluating all other variables at their values for each observation and then averaging. This yields the apparently surprising result that medical expenditures eventually decrease with age. This result arises because the fitted quadratic relationship  $1342.3 \times \text{age} - 9.452 \times \text{age}^2$  has a turning point at  $\text{age} = 1342.3/(2 \times 9.452) = 71.01$ , so `totexp` declines with `age` for  $\text{age} > 71$ . The raw data show expenditures decline in age beginning in the early 80s.

#### 4.4.6 Plots of predictive margins

The `marginsplot` command graphs the result of the immediately preceding `margins` command. The command options provide different ways of presenting the graphs, including plots, subgraphs, and graphs over groups of variables used in the preceding `margins` command. The default is to include 95% confidence intervals for the predictive mean in the graphs. The `noci` option omits these.

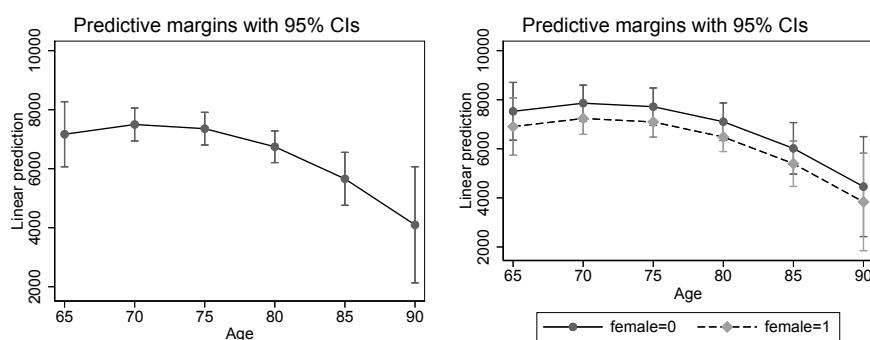
We first plot the overall predictive margin as age varies in 5-year increments from 65 to 90 years.

```

. * Marginsplot for predictive margins by age
. qui margins, at(age=(65(5)90))
. marginsplot
Variables that uniquely identify margins: age

```

The plot is given in the first panel of figure 4.1. As expected, the confidence intervals are widest at the sample extreme ages of 65 and 90.



**Figure 4.1.** Predictive margins 1) by age and 2) by age and gender

The `over(female)` option of the `margins` command leads to separate computation by gender of the predictive margins. We add this option to the preceding example with age varying in 5-year increments from 65 to 90 years.

```

. * Marginsplot for predictive margins by age for each gender
. margins, at(age=(65(5)90)) over(female) nofvlabel
Predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
Over:          female
1._at: 0.female
    age = 65
    1.female
    age = 65
2._at: 0.female
    age = 70
    1.female
    age = 70
3._at: 0.female
    age = 75
    1.female
    age = 75
4._at: 0.female
    age = 80
    1.female
    age = 80
5._at: 0.female
    age = 85
    1.female
    age = 85
6._at: 0.female
    age = 90
    1.female
    age = 90

```

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
_at#female						
1 0	7529.769	600.8899	12.53	0.000	6351.579	8707.958
1 1	6906.739	596.8414	11.57	0.000	5736.487	8076.99
2 0	7861.037	376.1116	20.90	0.000	7123.58	8598.495
2 1	7238.007	330.6136	21.89	0.000	6589.76	7886.255
3 0	7719.705	388.9309	19.85	0.000	6957.113	8482.298
3 1	7096.675	317.0534	22.38	0.000	6475.016	7718.335
4 0	7105.773	390.0599	18.22	0.000	6340.966	7870.58
4 1	6482.743	302.5363	21.43	0.000	5889.548	7075.938
5 0	6019.24	535.7552	11.24	0.000	4968.763	7069.717
5 1	5396.21	474.6123	11.37	0.000	4465.618	6326.802
6 0	4460.106	1039.299	4.29	0.000	2422.31	6497.903
6 1	3837.076	1012.879	3.79	0.000	1851.084	5823.069

```
. marginsplot
```

Variables that uniquely identify margins: age female

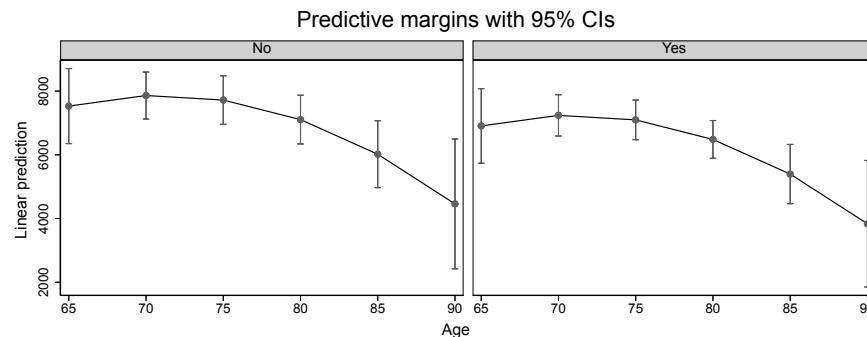
The plot is given in the second panel of figure 4.1. The 95% confidence intervals are overlapping, suggesting no statistically significant difference by gender at each age.

The confidence intervals for the different genders can be more clearly seen by using the `by(female)` option of the `marginsplot` command to produce separate graphs by gender.

```
. * Marginsplot for predictive margins by age, separate graphs by gender
. qui margins, at(age=(65(5)90)) over(female)
. marginsplot, by(female)
```

Variables that uniquely identify margins: age female

Figure 4.2 presents side-by-side graphs by gender that were combined in the second panel of figure 4.1.



**Figure 4.2.** Predictive margins 1) by age for men and 2) by age for women

#### 4.4.7 Pairwise comparisons of predictive margins

The `margins` command can produce several predictive margins. Interest may lie in calculating differences across these predictive margins and testing whether the differences are statistically significant.

The `margins, pwcompare` command performs pairwise comparisons across the levels of factor variables or variables defined by the `at()` option or `over()` option from the most recent `margins` command. If the `margins` command yields  $K$  predictive margins, then there are  $K!/\{2!(K - 2)!\}$

unique pairwise comparisons. The default output includes the pairwise differences in predictive margins and associated standard errors and confidence intervals.

As an example, consider comparison of the PM evaluated at variable `income` equaling 10, 20, and 30 (recall that `income` is measured in thousands of dollars per year).

We begin by obtaining the PMS at the various values of `income`.

```
. * Predictive margins for three income values
. margins, at(income=(10(10)30))

Predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
1._at: income = 10
2._at: income = 20
3._at: income = 30
```

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
_at						
1	6915.386	227.0065	30.46	0.000	6470.285	7360.487
2	7025.69	202.6934	34.66	0.000	6628.261	7423.12
3	7135.994	211.7422	33.70	0.000	6720.823	7551.166

There are then three unique pairwise comparisons: `income` 20 versus 10, `income` 30 versus 10, and `income` 30 versus 20. For each comparison, the `pwcompare` option default yields the difference in PMS and its standard error and associated 95% confidence interval. The `pwcompare(effects)` option additionally reports a *t*-test statistic and its associated degrees of freedom.

```

. * Pairwise comparison of predictive margins for three income values
. margins, at(income=(10(10)30)) pwcompare(effects)
Pairwise comparisons of predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
1._at: income = 10
2._at: income = 20
3._at: income = 30

```

	Contrast	Delta-method		Unadjusted		Unadjusted	
		std. err.	t	P> t	[95% conf. interval]		
_at							
2 vs 1	110.3041	84.25407	1.31	0.191	-54.89631	275.5045	
3 vs 1	220.6082	168.5081	1.31	0.191	-109.7926	551.009	
3 vs 2	110.3041	84.25407	1.31	0.191	-54.89631	275.5045	

From the preceding command, the predictive margins were 6915.39 for `income` 10, 7025.69 for `income` 20, and 7135.99 for `income` 30. Thus, the first contrast equals  $7025.69 - 6915.39 = 110.30$ , and so on. The output gives the three contrasts and additionally provides standard errors, a  $t$ -test statistic and its  $p$ -value, and a 95% confidence interval. For the first contrast,  $t = 110.30/84.25 = 1.31$ . At level 0.05, there is no statistically significant difference by `income` for any of the pairwise comparisons because all three test statistics have  $p > 0.05$ . The test statistics take the same value at each level of `income` because of the linearity of this example; more generally, they will differ.

For the (0, 1) binary indicator variable, the commands `margins female`, `pwcompare` and `margins, at(female=(0 1)) pwcompare` yield identical results. Both of these commands compute the margins with everyone treated as male and the margins with everyone treated as female and then compare the two margins. The command `margins, over(female) pwcompare` instead computes the margins for observations that are male in the dataset and the margins for observations that are female in the dataset and then compares the two margins.

A more complicated example compares income–gender combinations. Using three income categories and the two gender categories, we have

```

. * Pairwise comparison of predictive margins for six income-gender combinations
. margins female, at(income=(10(10)30)) pwcompare(effects) nofvlabel
Pairwise comparisons of predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
1._at: income = 10
2._at: income = 20
3._at: income = 30

```

	Delta-method		Unadjusted		Unadjusted	
	Contrast	std. err.	t	P> t	[95% conf. interval]	
<u>_at#female</u>						
(1 1) vs (1 0)	-1419.269	477.7069	-2.97	0.003	-2355.928	-482.6092
(2 0) vs (1 0)	20.90179	112.7985	0.19	0.853	-200.2669	242.0705
(2 1) vs (1 0)	-1244.128	457.1656	-2.72	0.007	-2140.511	-347.7445
(3 0) vs (1 0)	41.80358	225.5971	0.19	0.853	-400.5338	484.141
(3 1) vs (1 0)	-1068.987	465.6923	-2.30	0.022	-1982.089	-155.8848
(2 0) vs (1 1)	1440.171	445.1906	3.23	0.001	567.2671	2313.074
(2 1) vs (1 1)	175.1409	116.3454	1.51	0.132	-52.98229	403.2641
(3 0) vs (1 1)	1461.072	440.0368	3.32	0.001	598.274	2323.871
(3 1) vs (1 1)	350.2818	232.6908	1.51	0.132	-105.9646	806.5283
(2 1) vs (2 0)	-1265.03	421.6034	-3.00	0.003	-2091.685	-438.3746
(3 0) vs (2 0)	20.90179	112.7985	0.19	0.853	-200.2669	242.0705
(3 1) vs (2 0)	-1089.889	429.391	-2.54	0.011	-1931.813	-247.9641
(3 0) vs (2 1)	1285.931	414.663	3.10	0.002	472.8846	2098.978
(3 1) vs (2 1)	175.1409	116.3454	1.51	0.132	-52.98229	403.2641
(3 1) vs (3 0)	-1110.79	421.1068	-2.64	0.008	-1936.472	-285.109

There are six age–gender combinations and  $6!/(2!4!) = 15$  unique pairwise comparisons. The first comparison is that of (10, female) to (10, male) with 95% confidence interval  $[-2356, -483]$ . For those with an annual income of \$10,000, there is a statistically significant difference between female and male medical expenditures at level 0.05 because  $p = 0.003 < 0.05$ .

#### 4.4.8 Contrasts of predictive margins

As already noted, the `margins` command can produce several predictive margins. Interest may lie in calculating differences across these margins and testing whether they are statistically significant.

The `margins, contrast` command performs comparisons of predictive margins across the levels of factor categorical variables. These comparisons are done separately at each value of variables specified in the `at()` option or `over()` option, should these options be used. It can lead to comparison of

fewer contrasts than if the `margins`, `pwcompare` command is used, as illustrated below.

We first contrast predictive margins by gender. The `margins` command default is to report only *F*-test statistics of any contrasts. The `contrast(ci)` option additionally lists the difference in predictive margins. We have

```
. * Contrasts of predictive margins for gender
. margins female, contrast(ci) nofvlabel
Contrasts of predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()



|             | df   | F    | P>F    |
|-------------|------|------|--------|
| female      | 1    | 8.70 | 0.0032 |
| Denominator | 3054 |      |        |


|                       | Delta-method |           |                      |
|-----------------------|--------------|-----------|----------------------|
|                       | Contrast     | std. err. | [95% conf. interval] |
| female<br>(1 vs base) | -1226.86     | 415.9167  | -2042.365 -411.3547  |


```

From output given earlier, the predictive margins (with other variables set to their sample values) were 7764.12 for `female = 0` and 6537.26 for `female = 1`, leading to a contrast of  $6537.26 - 7764.12 = -1226.86$ . The `contrast(ci)` option reports this contrast and its standard error of 415.92. This leads to a *t*-test statistic of  $-1226.86/415.92 = -2.950$  and hence the *F* statistic of  $(-2.950)^2 = 8.70$ , also given in the output.

Note that in this simple example, there is only one possible pairwise comparison, between male and female. Consequently, the `margins female, pwcompare(ci)` command will list the same results.

Now consider contrasts of predictive margins by gender, at each of the three levels of `income` (10, 20, and 30), using `contrast(nowald)` to suppress the table of Wald tests. We have

```

. * Contrasts of predictive margins for gender at each of three income levels
. margins female, at(income=(10(10)30)) contrast(nowald) nofvlabel
Contrasts of predictive margins                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
1._at: income = 10
2._at: income = 20
3._at: income = 30

```

	Delta-method			
	Contrast	std. err.	[95% conf. interval]	
female@_at				
(1 vs base) 1	-1419.269	477.7069	-2355.928	-482.6092
(1 vs base) 2	-1265.03	421.6034	-2091.685	-438.3746
(1 vs base) 3	-1110.79	421.1068	-1936.472	-285.109

The first contrast is of `female = 1` against `female = 1` at `income = 10`. The difference in predictive margins is  $-1419.269$  and is statistically significant at level 0.05 because  $t = -1419.269/477.7069 = -2.97$ . This same result was obtained earlier using the `pwcompare(effects)` option. Then the difference (1 1) versus (1 0) equaled  $-1419.269$  with standard error 477.71 and  $p = 0.003$ . However, in the current `contrast` example, only 3 of the 15 pairwise contrasts of the `pwcompare` example appear, namely, (1 1) versus (1 0), (2 1) versus (2 0), and (3 1) versus (3 0).

The `contrast` and `margins, contrast` commands allow many different comparisons at various levels of nesting; see the Stata documentation. For simpler examples in which each explanatory variable changes in isolation, it can be easier to use the `margins, dydx()` command, which we present next.

## 4.5 Marginal effects

A ME, or partial effect, most often measures the effect on the conditional mean of  $y$  of a change in one of the explanatory variables, say  $x_j$ , when all other variables are unchanged. A brief introduction was given in section [3.5.9](#); further detail is provided here.

In a model with both parameters and regressors entering linearly, the ME equals the relevant slope coefficient, greatly simplifying analysis. An example is the model  $E(y|x) = \beta_1 + \beta_2x$ , for which the ME is simply  $\beta_2$ .

For models with nonlinearity in either parameters or explanatory variables, the ME may no longer equal the slope coefficient. Examples include the quadratic model  $E(y|x) = \beta_1 + \beta_2x + \beta_3x^2$ , the interactive model  $E(y|x, z) = \beta_1 + \beta_2x + \beta_3x \times z$ , and the exponential model  $E(y|x) = \exp(\beta_1 + \beta_2x)$ .

There are then two complications. First, the ME depends on the particular values of the regressors at which it is computed. Second, calculus and finite-difference methods lead to different values of the ME. There are many ways to compute MES, and the researcher should explicitly state the method used.

### 4.5.1 Calculus and finite-difference methods

Using calculus methods, we see the ME of the  $j$ th explanatory variable is

$$\text{ME}_j = \frac{\partial E(y|\mathbf{x} = \mathbf{x}^*)}{\partial x_j}$$

For the linear model with  $E(y|x) = \beta_1 + \beta_2x$ , we obtain simply  $\text{ME}_x = \beta_2$ . For the quadratic model with  $E(y|x) = \beta_1 + \beta_2x + \beta_3x^2$ , however, we obtain  $\text{ME}_x = \beta_2 + 2\beta_3x$ . This ME is not simply the relevant parameter  $\beta_2$  (or  $\beta_3$ ), and it varies with the point of evaluation  $x^*$ .

Calculus methods can always be applied but are not always the most appropriate method. In particular, for a  $(0, 1)$  indicator variable, we are interested in the change in the conditional mean when the indicator variable changes from 0 to 1. Or for the continuous variable age, we may be interested in the impact of age increasing from 40 to 60.

Let  $\mathbf{x}' = (z \ \mathbf{w}')$ , where interest lies in the ME of the scalar variable  $z$  and  $\mathbf{w}$  denotes all regressors other than  $z$ . Then the finite difference or discrete difference when  $z$  changes from  $z_1$  to  $z_2$  is

$$\Delta E(y) = E(y|\mathbf{w} = \mathbf{w}^*, z = z_2) - E(y|\mathbf{w} = \mathbf{w}^*, z = z_1)$$

For a  $(0, 1)$  indicator variable, this is simply  $E(y|\mathbf{w}^*, z = 1) - E(y|\mathbf{w}^*, z = 0)$ . More generally, for a set of indicators for  $K$  categories, the comparison made is between each category and the base category. For continuous regressors, a common discrete change to consider is an increase of one standard deviation from the sample mean value of the regressor of interest.

For the linear regression model with explanatory variables entering linearly, calculus and finite-difference methods give exactly the same result. For example, if  $E(y|x) = \beta_1 + \beta_2 x$  then the finite-difference method with a one-unit change from  $x^*$  to  $x^* + 1$  yields ME  
 $= \beta_2\{(x^* + 1) - x^*\} = \beta_2 = \partial E(y|x)/\partial x$ . If explanatory variables enter nonlinearly, then this is no longer the case. For example, if  $E(y|x) = \beta_1 + \beta_2 x + \beta_3 x^2$  then the finite-difference method with a one-unit change from  $x^*$  to  $x^* + 1$  yields ME  
 $= \{\beta_2(x^* + 1) + \beta_3(x^* + 1)^2\} - \{\beta_2x^* + \beta_3x^{*2}\} = \beta_2 + 2\beta_3x^* + \beta_3^2$ , whereas calculus methods evaluated at  $x^*$  yield ME =  $\beta_2 + 2\beta_3x^*$ .

For intrinsically nonlinear regression models, such as logit and Poisson models, calculus and finite-difference methods always give different results; see section [13.7](#). If interest lies in a unit change in the regressor, then in nonlinear models calculus methods provide only an approximation.

### 4.5.2 Average marginal effect, ME at mean, and ME at a representative value

Using calculus methods, we see the estimated ME of a change in variable  $x_j$ , evaluated at  $\mathbf{x}_i = \mathbf{x}_i^*$ ,  $i = 1, \dots, N$ , is computed as

$$\text{ME}_j = \frac{1}{N} \sum_{i=1}^N \left. \frac{\partial \widehat{E}(y_i | \mathbf{x}_i = \mathbf{x}_i^*)}{\partial x_{ij}} \right|_{\mathbf{x}_i = \mathbf{x}_i^*}$$

where  $\widehat{E}(y_i | \mathbf{x}_i = \mathbf{x}_i^*)$  is the fitted value when  $\mathbf{x}_i = \mathbf{x}_i^*$ . More generally, MES for the quantity  $g(\mathbf{x}, \boldsymbol{\beta})$  are computed by replacing  $\widehat{E}(y_i | \mathbf{x}_i)$  with  $g(\mathbf{x}_i, \widehat{\boldsymbol{\beta}})$ .

Three common choices of evaluation of  $\mathbf{x}_i^*$  are 1) at sample values; 2) at the sample mean of the regressors; and 3) at representative values of the regressors. In the last two cases, there is no need to average because  $\mathbf{x}_i^*$  takes only one value. We use the following acronyms, where the first two follow [Bartus \(2005\)](#).

AME	Average marginal effect	Average of MES at $\mathbf{x}_i^* = \mathbf{x}_i$
MEM	Marginal effect at mean	ME at $\mathbf{x}_i^* = \bar{\mathbf{x}}$
MER	Marginal effect at a representative value	ME at $\mathbf{x}_i^* = \mathbf{x}^*$

In most examples in this book, we compute the in-sample AME because this provides simple interpretation of results obtained from fitting models with nonlinear effects. In a nonlinear model, the average response, measured by the AME, differs from the response of the average individual, measured by the MEM. In specific applications, the MEM or the MER may be of more immediate interest than the AME.

The three quantities can be computed using the `margins` postestimation command with the `dydx()` option. Combinations of these methods are also possible. For example, several regressors may be set to a representative value, while the remaining regressors are evaluated at sample values, and

then the average is taken. The Stata documentation refers to such hybrids as conditional MES, rather than AMES.

In the special case that all variables are set to a single value (representative or mean) for all individuals, the evaluation can be done once. There is no need to compute for all individuals and average.

Using the finite-difference method, we see the estimated ME of a change in scalar variable  $z$  from  $z_1$  to  $z_2$  evaluated at  $\mathbf{w}_i = \mathbf{w}_i^*$ ,  $i = 1, \dots, N$ , is computed as

$$\text{ME}_j = \frac{1}{N} \sum_{i=1}^N \left\{ \widehat{E}(y_i | \mathbf{w}_i^*, z = z_2) - \widehat{E}(y | \mathbf{w}_i^*, z = z_1) \right\}$$

### 4.5.3 The margins, dydx() command

MES can be obtained with the `margins, dydx(varlist)` command, where `varlist` is a list of variables. The `margins, dydx(*)` command computes MES for all explanatory variables. As with the basic `margins` command (see section [4.4.2](#)), evaluation can be at specified values of some of or all the explanatory variables (the `at()` option) or at mean values of all explanatory variables (the `atmeans` option). The default is to estimate at sample values of variables. The `margins, dydx()` command supersedes Stata's `mfx` command and the community-contributed `margeff` command.

Thus, if interest lies in the average response across the sample, one computes the AME using the simple command `margins, dydx(*)`.

And if instead the interest lies in the response of the average individual in the sample, one computes the MEM using the command `margins, dydx() atmeans`.

The default is to compute MES using calculus methods for all variables, other than variables identified as factor-indicator variables, for which the finite-difference method is used. The option `continuous` instead also uses calculus methods for factor-indicator variables.

The default standard errors for estimated MES treat the regressors as fixed. If a population ME is desired and sample values of regressors are used in computing the ME, then the `vce(unconditional)` option additionally controls for variation in the regressors due to sampling; see section [13.7.9](#).

If population AMES are desired and population weights are provided, then one needs to compute weighted margins. If `weight` was used in the preceding estimation command, then weighted MES are automatically given. If the preceding estimation command did not use weights, as is commonly the case in microeconomics, but weighted MES are desired, then one needs to subsequently add `weight` to the `margins, dydx()` command.

#### 4.5.4 Average marginal effects

If explanatory variables enter nonlinearly, such as quadratically or interactively, then the original estimation command needs to enter those variables using factor-variable operators. For completeness, we use factor-variable operators to define all variables in the original OLS regression.

The following example computes AMES for all regressors in the same regression model as that used in the previous section.

```
. * AMEs computed with factor variables used to define model
. qui regress totexp i.suppins i.phylim i.actlim c.totchr c.age##c.age
> i.female##c.income, vce(robust) noheader
. margins, dydx(*) nofvlabel
Average marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
dy/dx wrt: 1.suppins 1.phylim 1.actlim totchr age 1.female income
```

	Delta-method					
	dy/dx	std. err.	t	P> t	[95% conf. interval]	
1.suppins	854.581	413.6253	2.07	0.039	43.56894	1665.593
1.phylim	2432.789	536.4219	4.54	0.000	1381.004	3484.573
1.actlim	3779.804	690.7463	5.47	0.000	2425.429	5134.178
totchr	1921.016	180.5999	10.64	0.000	1566.906	2275.125
age	-59.86769	39.07924	-1.53	0.126	-136.492	16.75658
1.female	-1226.86	415.9167	-2.95	0.003	-2042.365	-411.3547
income	11.03041	8.425407	1.31	0.191	-5.489631	27.55045

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

For the factor-indicator variables `suppins`, `phylim`, `actlim`, and `female` the AME is computed using a one-unit change from 0 to 1. For the continuous variables `totchr`, `age`, and `income`, calculus methods are used. The AME for `totchr`, which is continuous and does not appear interactively, is computed using calculus methods regardless of whether it is entered as `totchr` or as `c.totchr`.

Because the variables `suppins`, `phylim`, `actlim`, and `totchr` enter noninteractively, the AMEs for these variables equal the OLS slope coefficients given in the preceding section. For the remaining variables `age`, `female`, and `income` that appear interactively, the computation of AMEs requires additional analysis.

The following example manually computes the same AMEs for variables `age`, `female`, and `income`. We use

$$\begin{aligned} ME_{age} &= \beta_{age} + 2\beta_{age-squared} \times age \\ ME_{female} &= \beta_{female} + \beta_{female \times income} \times income \\ ME_{income} &= \beta_{income} + \beta_{female \times income} \times female \end{aligned}$$

```
. * AMEs computed manually
. generate meage = _b[age] + 2*_b[c.age#c.age]*age
. generate mefemale = _b[1.female] + _b[1.female#c.income]*income
. generate meincome = _b[income] + _b[1.female#c.income]*female
. sum meage mefemale meincome
```

Variable	Obs	Mean	Std. dev.	Min	Max
meage	3,064	-59.86769	120.4742	-359.0868	113.5138
mefemale	3,064	-1226.86	347.5765	-1588.932	3245.848
meincome	3,064	11.03041	7.614758	2.090179	17.51409

The complex names for stored coefficients such as `_b[1.female#c.income]` were obtained by first using the `coflegend` option of the `regress` command or by giving the command `matrix list e(b)` following the `regress` command.

The resulting AMEs are the same as those obtained more simply using the `margins, dydx(*)` command. Besides simplicity, the `margins, dydx(*)`

command has the advantage of reporting valid standard errors for the AMES.

For factor-indicator variables, the `margins, dydx(*)` command uses the finite-difference method. Finite differences can equivalently be calculated using the `margins, contrast` command. In fact, from the previous section, the command `margins female, contrast(ci)` calculated the same difference of  $-1226.86$  with the same standard error of  $415.9167$ . The `margins, contrast` command allows comparisons of PMS when combinations of variables change, whereas `margins, dydx(*)` allows only isolated changes in each variable.

What if the `margins, dydx(*)` command was used without first using factor variables to define the model? The following example does this, where first we need to generate variables for the quadratic regressor `age2` and the interactive regressor `female × income`. We have

```
. * Wrong way to compute (average) MEs if any nonlinearity present
. generate agesq = age^2
. generate fembyinc = female*income
. qui regress totexp suppins phylim actlim totchr age agesq
>     female income fembyinc, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,064
Model VCE: Robust

Expression: Linear prediction, predict()
dy/dx wrt: suppins phylim actlim totchr age agesq female income fembyinc
```

	Delta-method					
	dy/dx	std. err.	t	P> t	[95% conf. interval]	
suppins	854.581	413.6253	2.07	0.039	43.56894	1665.593
phylim	2432.789	536.4219	4.54	0.000	1381.004	3484.573
actlim	3779.804	690.7463	5.47	0.000	2425.429	5134.178
totchr	1921.016	180.5999	10.64	0.000	1566.906	2275.125
age	1342.275	763.421	1.76	0.079	-154.5957	2839.146
agesq	-9.452011	5.041687	-1.87	0.061	-19.33745	.433432
female	-1573.508	573.3188	-2.74	0.006	-2697.638	-449.3781
income	2.090179	11.27985	0.19	0.853	-20.02669	24.20705
fembyinc	15.42391	15.81709	0.98	0.330	-15.58931	46.43713

The underlying OLS estimates are suppressed but are the same as before. But now the AMES for variables `age`, `female`, and `income` are markedly different, and there are additional AMES for the regressors `agesq` and `fembyinc`. This is

the wrong approach. For example, the AME for `age` is computed assuming `agesq` does not change, even though `age` has changed.

At the same time, the AMEs are unchanged for the first four variables that did not appear nonlinearly via a quadratic or interactive term. This equivalence for the categorical regressors `suppins`, `phylim`, and `actlim` is a special property of a linear model because then the finite-difference and calculus methods give the same ME. Different MES would arise for a nonlinear model such as the logit model.

#### 4.5.5 Marginal effect at mean

The MEM is conceptually quite different from the AME because the ME for the average individual in general does not equal the average of the MES for each individual.

The two are equal in a purely linear model. They are also equal in this particular example because of the particular form of nonlinearity specified. Consider a model with quadratic terms and interactions with

$E(y|x) = \beta_1 + \beta_2x + \beta_3x^2 + \beta_4z + \beta_5z^2 + \beta_6xz$ . Then ME  
 $x = \beta_2 + 2\beta_3x + \beta_6z$ . So AME  
 $x = (1/N) \sum_{i=1}^N \hat{\beta}_2 + 2\hat{\beta}_3x_i + \hat{\beta}_6z_i = \hat{\beta}_2 + 2\hat{\beta}_3\bar{x} + \hat{\beta}_6\bar{z}$ . But this last quantity equals  $\text{MEM}_x$ . Similarly,  $\text{AME}_z = \text{MEM}_z$ . Note that there would be a difference if we used finite-difference rather than calculus methods.

Thus, in this example, `margins, dydx(*) atmeans` gives exactly the same estimates and standard errors as already obtained using command `margins, dydx(*)`.

#### 4.5.6 Marginal effect at a representative value

The following example computes the MER for each variable when all variables are set to specified values for a woman aged 70 years with supplemental insurance, physical and activity limitation, two chronic conditions, and annual income of \$20,000.

```

. * MER computed with factor variables
. qui regress totexp i.suppins i.phylim i.actlim c.totchr c.age##c.age
>      i.female##c.income, vce(robust) noheader
. margins, dydx(*) at(suppins=1 phylim=1 actlim=1 totchr=2 age=70 female=1
>      income=20) nofvlabel
Conditional marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
dy/dx wrt: 1.suppins 1.phylim 1.actlim totchr age 1.female income
At: suppins = 1
    phylim = 1
    actlim = 1
    totchr = 2
    age    = 70
    female = 1
    income = 20

```

	Delta-method					
	dy/dx	std. err.	t	P> t	[95% conf. interval]	
1.suppins	854.581	413.6253	2.07	0.039	43.56894	1665.593
1.phylim	2432.789	536.4219	4.54	0.000	1381.004	3484.573
1.actlim	3779.804	690.7463	5.47	0.000	2425.429	5134.178
totchr	1921.016	180.5999	10.64	0.000	1566.906	2275.125
age	18.99368	67.30506	0.28	0.778	-112.9741	150.9615
1.female	-1265.03	421.6034	-3.00	0.003	-2091.685	-438.3746
income	17.51409	11.63454	1.51	0.132	-5.298229	40.32641

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

The MER for the variables `age`, `female`, and `income` that entered interactively differ from the AMES presented earlier. The MES equal the AMES for the first four variables that did not appear nonlinearly via a quadratic or interactive term.

More generally, for other estimation commands such as `logit`, the MERS would differ from the AMES for all variables in the regression.

#### 4.5.7 Elasticities

The preceding MES estimate the change in the conditional mean as explanatory variables change. These MES use changes in levels. In some cases, it may be more informative to instead estimate elasticities that use proportionate or percentage changes. For example, we may be interested in

the percentage change in medical expenditures in response to a 1% change in income.

Elasticities and semielasticities can be computed using the `eyex()`, `dyex()`, and `eydx()` options of the `margins` command. Again, these can be computed at specified values of the regressors or at sample values for each observation and then averaged. As explained below, it is better to evaluate elasticities and semielasticities at specified values of the regressors.

The elasticity of  $y$  with respect to  $x$  is  $\partial y / \partial x \times (x/y)$ . Because the elasticity can be rewritten as  $(\partial y / y) / (\partial x / x)$ , it is interpreted as the proportionate change in  $y$  divided by the proportionate change in  $x$  or, equivalently, as the percentage change in  $y$  in response to a 1% change in  $x$ . To compute the elasticity, we use the `eyex()` option of the `margins` command.

Let  $\mathbf{x}_i^*$  denote the value at which the  $i$ th observation is evaluated. For example, if all variables are evaluated at sample means, then  $\mathbf{x}_i^* = \bar{\mathbf{x}}$ , while if all variables are evaluated at their sample values, then  $\mathbf{x}_i^* = \mathbf{x}_i$ . Then, using calculus methods, we see the `eyex()` option of the `margins` command computes the elasticity with respect to variable  $x_j$  as

$$\hat{\epsilon}_j = \frac{1}{N} \sum_{i=1}^N \left. \frac{\partial \hat{y}_i}{\partial x_{ij}} \right|_{\mathbf{x}_i^*} \times \frac{x_{ij}^*}{\hat{y}_i}$$

where  $\hat{y}_i = \hat{E}(y_i | \mathbf{x}_i = \mathbf{x}_i^*)$ .

The following example computes the sample average elasticity of `totexp` with respect to variable `income`:

```

. * Sample average elasticity with respect to income
. margins, eyex(income)
Average marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
ey/ex wrt: income

```

	Delta-method					
	ey/ex	std. err.	t	P> t	[95% conf. interval]	
income	-.0474743	13.56076	-0.00	0.997	-26.63662	26.54167

On average, a 1% change in income is associated with a 0.047% reduction in medical expenditures. This estimate is very noisy with a very large standard error and is highly statistically insignificant.

The following code manually computes the same average elasticity.

```

. * Manual computation of sample average elasticity with respect to income
. predict yhat
(option xb assumed; fitted values)
. generate eincome = (_b[income]+_b[1.female#c.income]*female) * income / yhat
. sum eincome

```

Variable	Obs	Mean	Std. dev.	Min	Max
eincome	3,064	-.0474743	3.839305	-199.924	3.528567

Again, the average estimated elasticity is  $-0.04747$ . But note that it varies greatly across observations, ranging from an implausibly large negative  $-199.924$  to an implausibly large positive  $3.529$ . This is most likely due to division by  $\hat{y}_i$  when  $\hat{y}_i$  takes values close to zero.

It can therefore be better to compute the elasticity at specified values of regressors, either representative values or the mean value of regressors. For example,

```

. * Elasticity with respect to income evaluated at representative values
. margins, eyex(income) at(suppins=1 phylim=1 actlim=1 totchr=2 age=70
>     female=1 income=20)
Conditional marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
ey/ex wrt: income
At: suppins = 1
    phylim = 1
    actlim = 1
    totchr = 2
    age    = 70
    female = 1
    income = 20

```

	Delta-method				
	ey/ex	std. err.	t	P> t	[95% conf. interval]
income	.0294402	.0197906	1.49	0.137	-.0093641 .0682444

When evaluation is at the specified values of all regressors, a 1% increase in income is associated with a 0.0294% increase in medical expenditures. This estimate is much more precisely estimated than the preceding sample average elasticity, though it is still statistically insignificant at level 0.05.

Elasticities are not meaningful for categorical variables, because it is not meaningful to consider, for example, the effect of a 1% change in gender. Accordingly, the `eyex()` option is not available for factor-indicator variables.

#### 4.5.8 Semielasticities

For indicator variables such as `female` or `actlim`, it is more meaningful to consider the proportionate change in medical expenditures when there is a one-unit change in the indicator variable. Such semielasticities can be obtained using the `eydx()` option.

For example, doing so at specified values of the variables yields

```

. * Semielasticities: Proportionate change in y w.r.t. unit change in x
. margins, eydx(*) at(suppins=1 phyylim=1 actlim=1 totchr=2 age=70
> female=1 income=20) nofvlabel
Conditional marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
ey/dx wrt: 1.suppins 1.phyylim 1.actlim totchr age 1.female income
At: suppins = 1
    phyylim = 1
    actlim = 1
    totchr = 2
    age = 70
    female = 1
    income = 20

```

	Delta-method					
	ey/dx	std. err.	t	P> t	[95% conf. interval]	
1.suppins	.074535	.0360862	2.07	0.039	.0037794	.1452906
1.phyylim	.2287452	.0551654	4.15	0.000	.1205802	.3369102
1.actlim	.3822587	.069224	5.52	0.000	.2465284	.5179889
totchr	.1614558	.0186426	8.66	0.000	.1249024	.1980091
age	.0015964	.005679	0.28	0.779	-.0095386	.0127313
1.female	-.101041	.0338938	-2.98	0.003	-.1674981	-.034584
income	.001472	.0009895	1.49	0.137	-.0004682	.0034122

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

For example, those with an activity limitation have medical expenditures that are a proportion 0.38 larger, or 38% larger, than medical expenditures for those without an activity limitation. And a one-unit change in variable income (a \$1,000 increase in annual income) is associated with a  $100 \times 0.001472 = 0.1472\%$  increase in medical expenditures.

Interest may also lie in the impact on the level of the dependent variable of a proportionate change in a regressor. Such semielasticities can be obtained using the `dyex()` option, which computes for the  $j$ th variable the quantity  $\sum_{i=1}^N (\partial y_i / \partial x_{ij})|_{\mathbf{x}_i^*} \times \mathbf{x}_{ij}^*$ . For indicator variables such as `female`, it makes no sense to consider a proportionate change, and the `dyex()` option is not available for factor-indicator variables.

Continuing the current example, we have

```

. * Semielasticities: Change in y w.r.t. proportionate change in x
. margins, dyex(totchr age income) at(suppins=1 phylim=1 actlim=1 totchr=2
> age=70 female=1 income=20)
Conditional marginal effects                                         Number of obs = 3,064
Model VCE: Robust
Expression: Linear prediction, predict()
dy/ex wrt:  totchr age income
At: suppins = 1
     phylim = 1
     actlim = 1
     totchr = 2
     age    = 70
     female = 1
     income = 20

```

	Delta-method					
	dy/ex	std. err.	t	P> t	[95% conf. interval]	
totchr	3842.031	361.1998	10.64	0.000	3133.812	4550.251
age	1329.557	4711.354	0.28	0.778	-7908.188	10567.3
income	350.2818	232.6908	1.51	0.132	-105.9646	806.5283

For example, a one-unit proportionate change or 100% change in the number of chronic conditions is associated with a \$3,842 increase in medical expenditures. So a 1% change in the number of chronic conditions is associated with a \$38 increase in medical expenditures.

## 4.6 Regression decomposition analysis

Often, regression data cover more than one group. Examples of dichotomous groupings include wage or earnings data that cover both union and nonunion workers and earnings data that cover both male and female workers or natives and immigrants.

Group membership is often heterogeneous in terms of observable characteristics such as age, experience, educational attainment, and so forth. When there are significant differences in the average market outcomes of groups, one is often interested in quantifying the relative contributions of observable characteristics and unobservable factors to this difference. For example, to what extent can lower average earnings of women compared with men be explained by observable characteristics? In the regression context, such analysis is called regression decomposition analysis.

### 4.6.1 Oaxaca–Blinder decomposition

A well-established framework for carrying out regression decomposition analysis is the so-called Oaxaca–Blinder decomposition of earnings or log earnings. This decomposition has been applied to study male–female wage differentials and union–nonunion wage differentials and also more specifically used in the context of analysis of gender discrimination. See [Fortin, Lemieux, and Firpo \(2011\)](#) for a survey that covers older materials as well as many modern interpretations and extensions.

Decomposition analysis can be interpreted as a form of counterfactual analysis that compares an actual outcome with an alternative hypothetical outcome generated under a different scenario or assumptions, as is standard in the potential-outcome model framework presented in section 24.2.

For specificity, we consider a standard linear regression model. Let  $y_i^g$  denote the outcome variable, here the natural logarithm of annual earnings, for individual  $i$  in group  $g$ . And let  $x_{ik}^g (i = 1, \dots, N_g; k = 1, \dots, K)$  denote the  $k$ th regressor for individual  $i$  in group  $g$ . For simplicity, assume two groups, here male and female workers with  $g = F$  or  $M$ . The sample

consists of observations on  $(y_i^g, x_{i1}^g, \dots, x_{iK}^g)$ . The regression equation of interest is

$$y_i^g = \beta_0^g + \sum_{k=1}^K x_{ik}^g \beta_k^g + u_i^g, \quad g = M, F$$

which allows all coefficients in the regression to differ between groups.

The average group differential is  $\hat{\Delta} = \bar{y}^F - \bar{y}^M$ , where  $\bar{y}^g$  is the group sample average. Under the assumption that  $\{E(u_i^F | \mathbf{x}_i^F) - E(u_i^M | \mathbf{x}_i^M)\} = 0$ , which means that any potential selection effect is the same for the two groups, the sample estimate of the average male–female differential is estimated to be

$$\hat{\Delta} = \bar{y}^F - \bar{y}^M = \left( \hat{\beta}_0^F + \sum_{k=1}^K \bar{x}_k^F \hat{\beta}_k^F \right) - \left( \hat{\beta}_0^M + \sum_{k=1}^K \bar{x}_k^M \hat{\beta}_k^M \right)$$

Two alternative Oaxaca–Blinder decompositions corresponding to this difference are easily derived:

$$\begin{aligned} \hat{\Delta}_1 &= \left( \hat{\beta}_0^F - \hat{\beta}_0^M \right) + \sum_{k=1}^K (\bar{x}_k^F - \bar{x}_k^M) \hat{\beta}_k^M + \sum_{k=1}^K (\hat{\beta}_k^F - \hat{\beta}_k^M) \bar{x}_k^F \\ \hat{\Delta}_2 &= \left( \hat{\beta}_0^F - \hat{\beta}_0^M \right) + \sum_{k=1}^K (\bar{x}_k^F - \bar{x}_k^M) \hat{\beta}_k^F + \sum_{k=1}^K (\hat{\beta}_k^F - \hat{\beta}_k^M) \bar{x}_k^M \end{aligned}$$

The second term in each decomposition is interpreted as the component of the differential that is accounted for by the mean differences in the observable regressors. This measures the “explained” component of the difference. The first and third terms are attributed to unobserved differences between the male and female groups that lead to differences in their

regression coefficients. These terms measure the “unexplained” components of the decomposition.

Whether one uses  $\widehat{\Delta}_1$  or  $\widehat{\Delta}_2$  depends upon which group is viewed as the reference group and which group is viewed as the treatment group. Yet another alternative is to use the regression estimates  $\widehat{\beta}_k^P$  obtained from the pooled sample of observations in place of either  $\widehat{\beta}_k^M$  or  $\widehat{\beta}_k^F$  in the second term;  $\widehat{\beta}_k^P$  can be interpreted as a matrix-weighted average of  $\widehat{\beta}_k^M$  and  $\widehat{\beta}_k^F$ .

Separate groupwise regressions yield the point estimates of the regression coefficients. If point estimates and confidence intervals are both required for the mean differential, then the variance estimates of sums of product terms like  $(\bar{x}_k^F - \bar{x}_k^M)\widehat{\beta}_k^M$  and  $(\widehat{\beta}_k^F - \widehat{\beta}_k^M)\bar{x}_k^F$  also have to be computed. This step is usually based on linearized approximations such as the delta method. For additional detail, see [Fortin, Lemieux, and Firpo \(2011\)](#). Many additional complications arise in practice, such as endogenous regressors and regressors that are defined only for one of the groups.

#### 4.6.2 An empirical example

In this empirical example, we use a subset of data from the Australian longitudinal survey, Medicine in Australia: Balancing Employment and Life. This survey provides information on the annual earnings of physicians in general practice. The dependent variable `logyearn` is the natural logarithm of annual earnings. The regressors are annual hours worked (`yhrs`); years of medical practice experience (`expr`) and its square (`exprsq`); fellowship of colleges and number of other postgraduate qualifications (`fellow` and `pgradoth`), which are two measures of professional qualifications; the number of full-time or part-time doctors in a practice (`pracsiz`); presence in home of a child under five years of age (`childu5`); and an indicator of temporary resident visa (`visa`).

Before we carry out a decomposition analysis, it is straightforward to check for differences in the two groups using a two-sample  $t$  test of difference in means. For example, for the dependent variable `logyearn`, we obtain

```
. * t tests on difference in mean log-earnings by gender
. qui use mus204mabeldecomp, clear
. ttest logyearn, by(female) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf. interval]
Male	8,970	12.47267	.0074711	.707592	12.45802 12.48731
Female	5,060	11.86746	.0098909	.7035751	11.84807 11.88685
Combined	14,030	12.25439	.0064466	.7635899	12.24176 12.26703
diff		.605207	.0123955		.5809096 .6295045

```
diff = mean(Male) - mean(Female) t = 48.8249
H0: diff = 0 Satterthwaite's degrees of freedom = 10542.9
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(T < t) = 1.0000 Pr(|T| > |t|) = 0.0000 Pr(T > t) = 0.0000
```

The null hypothesis of equality of means is easily rejected. There is a large gender difference of 0.61 in the average of `logyearn`. (Separate analysis of the earnings data in levels finds that average male earnings are 80% higher than average female earnings.)

Part of this large earnings differences may be due to differences in average characteristics by gender. In particular, male doctors annually average 2,377 annual hours worked versus 1,781 hours for female doctors.

### Gender difference in regressors

We first consider gender differences in the averages of the regressors. Using the expanded `table` command introduced in Stata 17, we obtain

```

. * Gender differences in variables for the regression estimation sample
. global xlist1 yhrs expr exprsq fellow pgradoth pracsize childu5 visa
. qui regress logyearn $xlist1
. table () (female) if e(sample), stat(mean logyearn $xlist1)
>      stat(count logyearn) nototals

```

	Male	Female
Mean		
log of annual earnings	12.1852	11.62744
Annual hours worked	2339.708	1661.385
Years of medical practice experience	26.18344	19.94557
expr squared	803.7421	481.9785
Fellowship of colleges	.5632425	.608076
Number of other postgrad qualifications	.546471	.6207443
Number of FT/PT doctors in practice: Bands	3.222572	3.426762
Have dep child udr 5y	.1439553	.175772
Temporary resident visa	.0230608	.0126683
Number of nonmissing values		
log of annual earnings	2,862	2,526

Because many variables have missing values, regression analysis uses a total of 5,388 observations, compared with 14,030 observations available for `logyearn`. On average, male doctors work 678 more hours annually and have 6.24 more years of experience, while women have somewhat higher professional qualifications.

#### Gender difference in regression coefficients

We run separate OLS regressions for male and female doctors and use the `estimates table` command to enable easy comparison of regression coefficients by gender.

```

. * Separate earnings regressions for male and female doctors
. qui regress logyearn $xlist1 if female==0, vce(robust)
. estimates store MALE
. qui regress logyearn $xlist1 if female==1, vce(robust)
. estimates store FEMALE
. estimates table MALE FEMALE, b(%11.5f) t(%11.2f) stats(N r2 F)

```

Variable	MALE	FEMALE
yhrs	0.00039 21.18	0.00056 27.07
expr	0.02452 6.35	0.00112 0.26
exprsq	-0.00050 -6.73	-0.00000 -0.02
fellow	-0.00457 -0.20	0.10117 4.53
pgradoth	0.03375 2.79	0.00276 0.20
pracsize	0.04418 4.82	0.00697 0.69
childu5	0.15863 5.43	0.02545 0.92
visa	-0.06945 -1.43	0.14580 2.06
_cons	10.85005 152.44	10.58729 144.30
N	2862	2526
r2	0.25497	0.38641
F	83.49789	108.14843

Legend: b/t

The two groups differ substantially in their sensitivity to annual hours worked, which is the most important and statistically significant driver of the differential. And the size of estimated coefficients of the other regressors and their statistical significance varies considerably by gender.

Formal tests of coefficient equality across gender require estimation of  $\text{Var}(\hat{\beta}_M - \hat{\beta}_F)$ . This depends in part on  $\text{Cov}(\hat{\beta}_M, \hat{\beta}_F)$ , which is not computed by the separate regressions by gender. So we use the `suest` command (see section [6.8.7](#)) to enable cross-equation inference.

```

. * Tests of coefficient equality in separate regressions
. qui regress logyearn $xlist1 if female==0
. estimates store MALE
. qui regress logyearn $xlist1 if female==1
. estimates store FEMALE
. suest MALE FEMALE

```

Simultaneous results for MALE, FEMALE Number of obs = 5,388

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
MALE_mean					
yhrs	.0003909	.0000184	21.21	0.000	.0003548 .000427
expr	.024517	.0038577	6.36	0.000	.0169561 .032078
exprsq	-.0004987	.000074	-6.74	0.000	-.0006437 -.0003537
fellow	-.0045728	.0226987	-0.20	0.840	-.0490615 .0399159
pgradoth	.0337479	.0120848	2.79	0.005	.0100621 .0574338
pracszie	.0441787	.0091453	4.83	0.000	.0262543 .0621032
childu5	.1586252	.0291587	5.44	0.000	.1014751 .2157753
visa	-.0694536	.0483732	-1.44	0.151	-.1642634 .0253562
_cons	10.85005	.0710696	152.67	0.000	10.71075 10.98934
MALE_lnvar					
_cons	-1.201407	.0429545	-27.97	0.000	-1.285597 -1.117218
FEMALE_mean					
yhrs	.0005571	.0000205	27.12	0.000	.0005168 .0005974
expr	.001117	.0043552	0.26	0.798	-.0074191 .0096532
exprsq	-2.35e-06	.0001024	-0.02	0.982	-.0002031 .0001984
fellow	.1011739	.0223206	4.53	0.000	.0574263 .1449215
pgradoth	.0027625	.0138797	0.20	0.842	-.0244411 .0299661
pracszie	.0069707	.0101001	0.69	0.490	-.0128252 .0267665
childu5	.0254493	.0277562	0.92	0.359	-.0289519 .0798504
visa	.1457976	.0707606	2.06	0.039	.0071093 .2844859
_cons	10.58729	.0732471	144.54	0.000	10.44372 10.73085
FEMALE_lnvar					
_cons	-1.391516	.0408461	-34.07	0.000	-1.471573 -1.311459

```

. test _b[MALE_mean:yhrs] = _b[FEMALE_mean:yhrs]
( 1) [MALE_mean]yhrs - [FEMALE_mean]yhrs = 0
      chi2( 1) =    36.26
      Prob > chi2 =    0.0000

```

The null hypothesis of gender equality of the coefficient of `yhrs` is strongly rejected. Note that `suest` by default gives heteroskedastic-robust standard errors, even though the individual `regress` commands did not specify the

`vce (robust)` option. The standard errors and  $t$  statistics differ slightly from those obtained using the individual `regress` commands because of slightly different degrees-of-freedom correction.

Given the difference in coefficients by gender, combined with the earlier observation that regressor averages vary by gender, we expect that a full decomposition will show that both the differences in regressors (“endowments”) and differences in coefficients will jointly account for the differences in average earnings of the two groups.

### **Regression decomposition estimates**

We could carry out the decomposition analysis by running separate regressions for male and female doctors and then building up the components of the differential. This will require some facility with matrix operations in Stata. Instead, we use the community-contributed command `oaxaca` ([Jann 2008](#)), which provides the decomposition in a more compact form. We obtain

```
. * Blinder-Oaxaca decomposition in logarithmic units
. oaxaca logyearn $xlist1, by(female) vce(robust)
```

Blinder-Oaxaca decomposition

	Number of obs = 5,388
	Model = linear
Group 1: female = 0	N of obs 1 = 2862
Group 2: female = 1	N of obs 2 = 2526

logyearn	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
<b>overall</b>						
group_1	12.1852	.0118725	1026.34	0.000	12.16193	12.20847
group_2	11.62744	.0126594	918.48	0.000	11.60263	11.65225
difference	.5577628	.0173556	32.14	0.000	.5237466	.5917791
endowments	.3786503	.0201843	18.76	0.000	.3390898	.4182107
coefficients	.317233	.0201198	15.77	0.000	.2777989	.3566672
interaction	-.1381205	.0227893	-6.06	0.000	-.1827868	-.0934542
<b>endowments</b>						
yhrs	.3778961	.0176741	21.38	0.000	.3432555	.4125366
expr	.006968	.0272152	0.26	0.798	-.0463728	.0603088
exprsq	-.0007548	.0330052	-0.02	0.982	-.0654437	.0639341
fellow	-.004536	.0016885	-2.69	0.007	-.0078454	-.0012265
pgradoth	-.0002052	.0010343	-0.20	0.843	-.0022324	.0018221
pracszie	-.0014233	.0020779	-0.68	0.493	-.005496	.0026493
childu5	-.0008097	.0009206	-0.88	0.379	-.0026141	.0009947
visa	.0015152	.0009029	1.68	0.093	-.0002545	.0032849
<b>coefficients</b>						
yhrs	-.2761344	.0459881	-6.00	0.000	-.3662693	-.1859995
expr	.4667255	.1163091	4.01	0.000	.2387638	.6946871
exprsq	-.239237	.0611112	-3.91	0.000	-.3590127	-.1194612
fellow	-.0643021	.0194158	-3.31	0.001	-.1023563	-.0262479
pgradoth	.0192341	.0114525	1.68	0.093	-.0032124	.0416805
pracszie	.1275032	.0467723	2.73	0.006	.0358313	.2191751
childu5	.0234086	.0071587	3.27	0.001	.0093778	.0374394
visa	-.0027269	.0011885	-2.29	0.022	-.0050562	-.0003975
_cons	.2627621	.1022213	2.57	0.010	.0624119	.4631122
<b>interaction</b>						
yhrs	-.1127423	.0190288	-5.92	0.000	-.150038	-.0754466
expr	.145966	.0369083	3.95	0.000	.0736269	.218305
exprsq	-.159712	.0412702	-3.87	0.000	-.2406001	-.0788239
fellow	.004741	.0020151	2.35	0.019	.0007915	.0086904
pgradoth	-.0023014	.0015225	-1.51	0.131	-.0052854	.0006826
pracszie	-.0075975	.0030317	-2.51	0.012	-.0135395	-.0016556
childu5	-.0042372	.0018513	-2.29	0.022	-.0078657	-.0006087
visa	-.002237	.0011792	-1.90	0.058	-.0045482	.0000741

The first segment of the output is most informative. The use of the log transformation results in the decomposition in terms of log earnings differences between the two groups. The decomposition shows the mean predicted log of yearly earnings for males and females, respectively, in the first two rows. The third row shows that the average differential is 0.5578. Of this differential, 0.3787 is accounted for by the differences in “endowments”, which refers to the mean differences in the regressors of the two groups, and a further 0.3172 is accounted for by the differences in the regression coefficients. The sum exceeds 0.5578; the difference of – 0.1381 is attributed to an interaction effect, the combined effect of the previous two effects. The components of the decomposition are quite precisely estimated, and both endowments and returns to the endowments matter.

The `eform` option of the `oaxaca` command executes a retransformation and allows the user to display the differential in natural units (Australian dollars).

The `eform` option exponentiates all the values in the preceding table. For example,  $\exp(0.5578) = 1.7468$ . This provides a good guide, but note that it does not control for the retransformation bias studied in section [4.2.3](#). From output not reproduced here, male earnings are overall 1.75 times female earnings (or 75% higher), with a multiple 1.46 attributed to gender differences in the levels of individual characteristics, such as hours worked, and a multiple 1.37 attributed to different returns to these individual characteristics.

## 4.7 Shapley decomposition of relative regressor importance

When using linear regression analysis, an investigator may be interested in considering the relative importance of the regressors. Given a list of regressors, say,  $x_1, \dots, x_K$ , is there a systematic procedure for quantifying their relative contributions to the goodness of fit, measured in terms of  $R^2$  or residual variance of the regression?

As a specific example, suppose the variable of interest is a measure of income inequality. Then the decomposition aims to establish the percentage contribution of regressors to the reduction in unexplained variance of the inequality measure. Neither significance tests nor stepwise regression provides a solution.

This decomposition problem is analogous to the “Shapley value decomposition” in cooperative game theory. In a case in which multiple inputs combine to produce some output, and inputs could be employed in any combination of inclusion and exclusion, then how are the contributions of individual inputs to be ranked?

When applied to a regression model with  $K$  regressors, this decomposition generates a ranking of the regressors in terms of their contributions to the reduction in unexplained variance or to the increase in  $R^2$ . This decomposition aims to measure the marginal contribution of factor  $x_k$ , which is the product of the ME and the change in  $x_k$ ,  $\beta_k \Delta x_k$ . The reduction in the unexplained variance,  $\sigma_y^2$ , may be expressed as  $\sigma_y^2 - \sigma_{y_k}^2$  — the difference between total unexplained variance and unexplained variance after adding  $x_k$  to the regression.

The marginal contribution of a variable depends upon the order in which the variable of interest is included in the regression. Thus, adding  $x_K$  to a regression with only  $x_1$  included will yield a different estimate of the marginal contribution to that obtained from the regression in which  $x_K$  is added to  $x_1$  and  $x_2$ . This issue is resolved by averaging the marginal contribution of a regressor over all  $K!$  unique regressions where the regressions include those with some of the regressors omitted; see [Shorrocks \(2013\)](#).

Implementation of the Shapley value calculation simply requires estimation of  $K!$  linear regressions followed by averaging of the estimated marginal impacts. This yields an estimate of expected marginal impact when the expectation is over all possible combinations of regressors. The result is expressed as a percentage increase in the regression  $R^2$ , or as a percentage decrease in residual variance, due to a particular factor.

A modified version of the Shapley decomposition is the Shapley–Owen decomposition, which allows one to group subsets of regressors and measure the group contribution to the improvement in unexplained variance. For example, if a variable enters as a quadratic, then both the variable and its square constitute one group.

#### 4.7.1 Empirical example

We continue with the same model as in the previous section and present the decomposition for the regression for men. Because the joint contribution of the experience variables `exp` and `exprsq` is of more interest than their separate contributions, we group these two variables. Similarly, the variables `fellow` and `pgradoth` are grouped.

We use the community-contributed `rego` command ([Huettner and Sunder 2012](#)).

```

. * Shapley-Owen decomposition of regressors' contributions to R-squared
. qui rego logyearn yhrs expr exprsq pracsize fellow pgradoth
>     if female==0, vce(robust)
. rego logyearn yhrs \ expr exprsq \ pracsize \ fellow pgradoth \
>     if female==0, vce(robust)

```

Gr	Regressor	Coef.	Std.Err.	P> t	Std.Coeff.	Group %R2
1	yhrs	.0003912 ***	.0000184	0.000	0.4412	78.5485
2	expr	.0172065 ***	.0036226	0.000	0.2943	16.8196
	exprsq	-.0004 ***	.0000721	0.000	-0.3742	
3	pracszie	.0451038 ***	.0090868	0.000	0.0896	2.2844
4	fellow	.0063228	.0221348	0.775	0.0049	2.3476
	pgradoth	.0361601 ***	.0121119	0.003	0.0447	
-	Intercept	10.97202	.0643553	0.000		
Observations						
Overall R2						
Root MSE						
F-stat. Model						
Log Likelihood						

Output is suppressed for the first command, which provides the Shapley decomposition when all regressors are treated as distinct. On execution, this would deliver the Shapley decomposition. The second command groups some of the variables and the last column of the table shows, for example, that the two experience variables jointly account for 16.8% of the fit.

By far the most important variable is annual hours worked (*yhrs*), which accounts for 78.55% of the explained variation.

## 4.8 Difference-in-differences estimators

A popular application of linear regression is the estimation of treatment effects using observational data or data from natural experiments. The data setup is one in which observations are available on treated and untreated groups before and after the application of treatment. A treatment variable undergoes a change due to change in policy or a natural experiment—a change that can be treated as an exogenous variation in a treatment variable.

To estimate the effect of the change, one can compare the outcome for the treated group with that for the untreated group. The treatment indicator is a single binary regressor that equals 1 if treatment occurs and equals 0 if treatment does not occur.

An important complication is that the outcome variable of interest may change even in absence of the treatment. That is, for both groups some change in the outcome is expected to take place over time even in the absence of treatment. Any attempt to estimate the treatment effect must control for these factors because otherwise the estimated treatment effects will be contaminated by other factors. The difference-in-differences estimator is one way to eliminate the effects of these other factors provided assumptions given below are satisfied; chapters 24 and 25 present other methods for evaluating the effect of a treatment.

We consider the before/after framework in which data are available on the treated and the comparison (control) groups both before (subscript  $b$ ) and after (subscript  $a$ ) the experiment. Let variable  $D$  take value 1 if treated and take value 0 otherwise. Then for the  $i$ th treated case, the change in the outcome over time is measured by  $(y_{ia} - y_{ib}|D_{ia} = 1)$ , and the change in the outcome for the untreated group is measured by  $(y_{ia} - y_{ib}|D_{ia} = 0)$ .

Then the *difference-in-differences measure*

$(y_{ia} - y_{ib}|D_{ia} = 1) - (y_{ia} - y_{ib}|D_{ia} = 0)$  forms the basis of an estimate of the treatment effect.

The validity of this method relies on certain assumptions. We suppose that earnings  $y_{it}$  for each individual are determined by the sum of an

individual-specific effect  $\phi_i$ , a time-specific term  $\delta_t$ , and an amount  $\alpha$  for those who receive treatment, where  $D_{it} = 1$  if treated. So

$$y_{it} = \phi_i + \delta_t + \alpha D_{it} + \varepsilon_{it}, \quad t = a, b$$

where  $\delta_t$  is a common “trend” effect assumed to apply to both groups.

Using the “before” and “after” formulation, and suppressing the model errors, we see  $y_{ia} - y_{ib} = \alpha + (\delta_a - \delta_b)$  if  $D_{it} = 1$  and  $y_{ia} - y_{ib} = (\delta_a - \delta_b)$  if  $D_{it} = 0$ . It follows that the treatment effect is given by

$$E(y_{ia} - y_{ib}|D_{ia} = 1) - E(y_{ia} - y_{ib}|D_{ia} = 0) = \alpha$$

Note that it is assumed that the trend over time,  $\delta_a - \delta_b$ , is assumed to be the same for the treated and untreated groups, a crucial assumption called the parallel trends assumption.

Replacing the expectations operator by the sample average, we see  $\alpha$  is simply the difference between average change in the outcome of the treated group and of the untreated group, that is,  $\hat{\alpha} = \Delta\bar{y}^{tr} - \Delta\bar{y}^{nt}$ . Any time-invariant factors, the potential confounders, are eliminated when we take a difference between the before and after outcomes. The assumption that the treatment effect and the confounding factors enter additively and separably makes it easier to eliminate the latter.

In this section, we consider the simplest case of treatment at the individual level and two time periods (before and after). Extensions to richer settings are presented in section 25.6.1.

#### 4.8.1 Example: Estimating the effect of training on earnings

As an illustration, we revisit the data from the National Supported Work (NSW) demonstration project, conducted in the 1970’s. The target parameter,

the impact of training on earnings, can be estimated by a randomized experiment that assigns some individuals to receive training (a treatment group) and others to receive no training (a control group). The effect of training could then be measured by direct comparison of sample means of posttreatment earnings for the treatment and control (reference) groups.

Those who receive treatment may differ from those who do not in terms of traits that are observable. Moreover, such observable factors may also directly affect the average outcome. Hence, a comparison of the treated with the nontreated must control for differences in observed characteristics and possibly in unobserved characteristics.

We use data from [Dehejia and Wahba \(1999\)](#) to illustrate the DID method. These data are a subset of data used in [LaLonde \(1986\)](#), who studied the outcomes of the NSW treated group and compared them with those of synthetic control groups drawn from two national surveys.

The treated sample is one of 185 males who received training during 1976–1977. The control group is one of 2,490 male household heads under the age of 55 who are not retired, drawn from the Panel Survey of Income Dynamics. [Dehejia and Wahba \(1999, 2002\)](#) call these two samples the  $\text{re}^{74}$  subsample (of the NSW treated) and the Panel Survey of Income Dynamics sample (of nontreated).

The treatment indicator variable  $D$  is defined as  $D = 1$  if training was received (so the observation is in the treated sample) and  $D = 0$  if no training was received (and the observation is in the control sample). The key variables are

```

. qui use mus204nswpsid, clear
. * DID: Outcome, treatment, and other variables
. describe re78 re75 treat age educ nodegree black hisp marr u74 u75 re74
>      agesq educsq re74sq re75sq u74black u74hisp

```

Variable name	Storage type	Display format	Value label	Variable label
re78	float	%9.0g		Real annual earnings in 1978 (posttreatment)
re75	float	%9.0g		Real annual earnings in 1975 (pretreatment)
treat	float	%9.0g		Treated
age	float	%9.0g		Age in years
educ	float	%9.0g		Years of education
nodegree	float	%9.0g		Years of education > 12
black	float	%9.0g		Black
hisp	float	%9.0g		Hispanic
marr	float	%9.0g		Married
u74	float	%9.0g		Unemployed in 1974
u75	float	%9.0g		Unemployed in 1975
re74	float	%9.0g		Real annual earnings in 1974 (pretreatment)
agesq	float	%9.0g		Age in years squared
educsq	float	%9.0g		Education in years squared
re74sq	float	%9.0g		Real annual earnings in 1974 squared
re75sq	float	%9.0g		Real annual earnings in 1975 squared
u74black	float	%9.0g		Unemployed and black
u74hisp	float	%9.0g		Unemployed and hispanic

The outcome of interest is posttreatment earnings, `re78`, the treatment variable is `treat`, and pretreatment earnings are `re75`.

The `table` command provides descriptive statistics sorted by treatment status.

```

. * Descriptive statistics for treatment and control samples
. table () (treat), stat(mean re78 re75 treat age educ nodegree black hisp
>     marr u74 u75 re74 agesq educsq re74sq re75sq u74black u74hisp)
>     stat(count re75) nototals

```

	Treated	
	0	1
Mean		
Real annual earnings in 1978 (posttreatment)	21553.92	6349.145
Real annual earnings in 1975 (pretreatment)	19063.34	1532.056
Treated	0	1
Age in years	34.8506	25.81622
Years of education	12.11687	10.34595
Years of education > 12	.3052209	.7081081
Black	.2506024	.8432432
Hispanic	.0325301	.0594595
Married	.8662651	.1891892
Unemployed in 1974	.0863454	.7081081
Unemployed in 1975	.1	.6
Real annual earnings in 1974 (pretreatment)	19428.75	2095.574
Age in years squared	1323.53	717.3946
Education in years squared	156.3161	111.0595
Real annual earnings in 1974 squared	5.57e+08	2.81e+07
Real annual earnings in 1975 squared	5.48e+08	1.27e+07
Unemployed and black	.0144578	.6
Unemployed and hispanic	.0036145	.0324324
Number of nonmissing values		
Real annual earnings in 1975 (pretreatment)	2,490	185

The treated group, the second group listed, differs considerably from the control group, being disproportionately black (84%) with less than high school degree (71%) and unemployed in the pretreatment year 1975 (71%). Estimates of the effect of training should control for these differences.

#### 4.8.2 Before–after comparison

One approach is to simply compute the change over time in the earnings for those receiving the treatment of training. This estimate is the mean difference in `re78` and `re75` for those with `treat=1`. From previous output, this equals  $(6349 - 1532) = 4817$ .

We have

```
. * Before--after comparison for the treated
. generate badiff = re78 - re75
. mean badiff if treat==1
```

Mean estimation		Number of obs = 185	
	Mean	Std. err.	[95% conf. interval]
badiff	4817.09	608.4203	3616.713 6017.467

There is a highly statistically significant effect of treatment.

The limitation of this approach is that earnings in general may also have increased over these three years for the nontreated.

#### 4.8.3 Treatment–control comparison

The outcome of interest is posttreatment earnings, `re78`, which were on average 6,349 in the treatment group and 21,554 in the control group. A simple estimate of the treatment effect is to compare average posttreatment earnings of treatment and control, leading to an estimate of  $6349 - 21554 = -15205$ .

This estimate can be computed using the difference-in-means-test command `ttest re78, by(treat) unequal`. Equivalently, it can be computed as the coefficient of the treatment indicator `treat` in OLS regression of `re78` on an intercept and `treat`. This is called a treatment–control comparison estimator because it mimics the analysis in an experimental setting. We have

. * Treatment-control comparison (difference in means)	
. regress re78 treat, vce(robust)	
Linear regression	Number of obs = 2,675
	F(1, 2673) = 537.36
	Prob > F = 0.0000
	R-squared = 0.0609
	Root MSE = 15152
<hr/>	
re78	Robust
	Coefficient std. err. t P> t  [95% conf. interval]
treat	-15204.78 655.9143 -23.18 0.000 -16490.93 -13918.63
_cons	21553.92 311.785 69.13 0.000 20942.56 22165.29

The estimated treatment effect of  $-15204.78$  seems large and has a perverse sign. We next consider the DID estimator of the treatment effect.

#### 4.8.4 Difference-in-differences estimate

The DID estimator of the average treatment effect on the treated (ATET) uses the difference across treatment and control groups of the change over time in average earnings. Average earnings in the treatment group grew by  $6349 - 1532 = 4817$ . Average earnings in the control group grew by  $21554 - 19063 = 2491$ . The DID estimate is then  $4817 - 2491 = 2326$ , a much more plausible estimate than using the simple comparison of posttreatment earnings.

Table 4.1 gives these computations. It includes an alternative ordering of the computation, first computing the difference in earnings between treated and control groups in each year and then computing the difference over years. The same result is obtained.

**Table 4.1.** DID computation example

	Treated	Not treated	Diff over treatment
Post (year = 2)	6,349	21,554	-15,205
Pre (year = 1)	1,532	19,063	-17,531
Change over time	4,817	2,491	DID = $4817 - 2491 = 2326$ or DID = $-15205 - (-17531) = 2326$

The DID estimator can be shown to be equivalent to the estimate of  $\alpha$  in the OLS regression

$$\begin{aligned} \text{re}_{it} &= \phi + \delta \times \text{dpost}_t + \gamma \times \text{dtreat}_i + \alpha \times (\text{dpost}_t \times \text{dtreat}_i) + u_{it} \\ i &= 1, \dots, 2675, \quad t = 75, 78 \end{aligned}$$

Here we have two periods of data, so there are now  $2 \times 2675 = 5350$  observations.  $\text{re}_{i,t}$  denotes earnings in the two periods, so it equals  $\text{re}_{i,75}$  in the pretreatment period and  $\text{re}_{i,78}$  in the posttreatment period, so the regression is one with 5,350 earnings observations. The indicator variable  $\text{dpost}_t$  equals one in the posttreatment period (1978) and equals zero in the pretreatment period (1975). The indicator variable  $\text{dtreat}_i$  equals one if the individual is in the treated sample and equals zero for the untreated. The interaction term  $\text{dpost}_t \times \text{dtreat}_i$  equals one for treated individuals in the posttreatment period; otherwise, it equals zero.

The advantage of running this regression is that it provides a standard error for treatment effect, enabling statistical inference. Furthermore, by using the `vce(cluster clustvar)` option, we can control for clustering due to the regression having two observations per individual.

To implement this regression, we need to convert the current dataset, which is in wide form, with a single observation having data for both years, to long form, with separate observations for each year. This is done using the `reshape` command (see section [8.10.2](#)) as follows.

```

. * DID - first stack into panel format
. generate id = _n
. generate earns1 = re75
. generate earns2 = re78
. qui reshape long earns, i(id) j(year)
. generate dpost = 0
. replace dpost = 1 if year==2
(2,675 real changes made)
. rename treat dtreat
. qui generate dpostbydtreat = dpost*dtreat
. summarize id year dtreat earns dpost dpostbydtreat age, sep(0)

```

Variable	Obs	Mean	Std. dev.	Min	Max
id	5,350	1338	772.2781	1	2675
year	5,350	1.5	.5000467	1	2
dtreat	5,350	.0691589	.2537478	0	1
earns	5,350	19176.63	14839.18	0	156653
dpost	5,350	.5	.5000467	0	1
dpostbydtreat	5,350	.0345794	.1827292	0	1
age	5,350	34.22579	10.49886	17	55

There are twice as many observations; half are in the postperiod (variable `dpost` has mean 0.5);  $185/2675 = 0.06916$  are treated; and  $0.5 \times 0.06916 = 0.03458$  are both treated and in the postperiod.

We then perform the DID regression:

```

. * DID estimation - no controls
. regress earns dpost dtreat dpostbydtreat, vce(cluster id)

Linear regression                               Number of obs     =      5,350
                                                F(3, 2674)      =     913.63
                                                Prob > F       =     0.0000
                                                R-squared       =     0.0867
                                                Root MSE        =    14185
(Std. err. adjusted for 2,675 clusters in id)

```

earns	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
dpost	2490.585	217.2256	11.47	0.000	2064.637	2916.532
dtreat	-17531.28	360.6329	-48.61	0.000	-18238.43	-16824.13
dpostbydtreat	2326.505	644.7524	3.61	0.000	1062.242	3590.769
_cons	19063.34	272.5572	69.94	0.000	18528.89	19597.78

This produces an estimated treatment effect (the coefficient of `dpostbydtreat`) of 2326.51. The treatment effect on the treated is highly statistically significant, with  $p = 0.000$ .

The regression formulation of DID allows the addition of control variables, such as `age`, to the regression. Thus, the intercept  $\phi$  is replaced by  $\mathbf{x}'_{it}\beta$ . We have

```
. * DID estimation - with time-invariant controls
. regress earns dpost dtreat dpostbydtreat age educ nodegree black hisp marr
>      u74 u75 re74 agesq educsq re74sq re75sq u74black u74hisp, vce(cluster id)
Linear regression
Number of obs      =      5,350
F(18, 2674)        =     904.53
Prob > F          =     0.0000
R-squared          =     0.7250
Root MSE           =    7795.2
(Std. err. adjusted for 2,675 clusters in id)
```

earns	Coefficient	Robust		t	P> t	[95% conf. interval]
		std. err.				
dpost	2490.585	217.531	11.45	0.000	2064.039	2917.13
dtreat	-2458.445	490.5804	-5.01	0.000	-3420.4	-1496.489
dpostbydtreat	2326.505	645.6588	3.60	0.000	1060.464	3592.546
age	31.63905	89.52798	0.35	0.724	-143.912	207.1901
educ	-411.9222	195.8797	-2.10	0.036	-796.0133	-27.8311
nodegree	209.4768	372.9206	0.56	0.574	-521.765	940.7186
black	-493.3401	268.5493	-1.84	0.066	-1019.925	33.24521
hisp	-15.52322	703.3069	-0.02	0.982	-1394.604	1363.557
marr	657.2537	302.36	2.17	0.030	64.3705	1250.137
u74	7273.687	1271.792	5.72	0.000	4779.893	9767.482
u75	-7980.635	809.0581	-9.86	0.000	-9567.078	-6394.192
re74	.8554493	.0853382	10.02	0.000	.6881138	1.022785
agesq	-.9252285	1.240772	-0.75	0.456	-3.358197	1.50774
educsq	36.02474	9.589573	3.76	0.000	17.22101	54.82846
re74sq	-9.28e-06	1.42e-06	-6.54	0.000	-.00000121	-6.50e-06
re75sq	9.37e-06	9.54e-07	9.81	0.000	7.49e-06	.00000112
u74black	1055.096	747.4911	1.41	0.158	-410.6231	2520.815
u74hisp	-273.6814	1479.979	-0.18	0.853	-3175.7	2628.337
_cons	1606.092	1764.901	0.91	0.363	-1854.617	5066.801

In this illustrative example, there is no change in the coefficients of `dpostbydtreat` and `dpost` because the additional control variables are all time invariant so that  $\mathbf{x}_{it} = \mathbf{x}_i$ . But the model fit is much better, with  $R^2$  increasing from 0.087 to 0.725. For similar reasons, there is little change in the standard error for `dpostbydtreat`.

Stata 17 introduced the `didregress` command for repeated cross-sections and the `xtdidregress` command for panel data. These commands, which can accommodate richer DID settings, are presented in section 25.6.

For the current panel-data example we apply `xtdidregress` and obtain

```
. * xtdidregress command for DID - no controls
. xtset id year

Panel variable: id (strongly balanced)
Time variable: year, 1 to 2
    Delta: 1 unit

. xtdidregress (earns) (dpostbydtreat), group(id) time(year)
Number of groups and treatment time

Time variable: year
Control:      dpostbydtreat = 0
Treatment:    dpostbydtreat = 1



|                 | Control | Treatment |
|-----------------|---------|-----------|
| Group<br>id     | 2490    | 185       |
| Time<br>Minimum | 1       | 2         |
| Maximum         | 1       | 2         |



Difference-in-differences regression                               Number of obs = 5,350
Data type: Longitudinal
(Std. err. adjusted for 2,675 clusters in id)



| earns                             | Robust      |           |      |       |                      |
|-----------------------------------|-------------|-----------|------|-------|----------------------|
|                                   | Coefficient | std. err. | t    | P> t  | [95% conf. interval] |
| ATET<br>dpostbydtreat<br>(1 vs 0) | 2326.505    | 644.6921  | 3.61 | 0.000 | 1062.36 3590.651     |


```

Note: ATET estimate adjusted for panel effects and time effects.

The variables provided in the command are, in order, 1) the outcome (and any control variables); 2) a treatment indicator equal to 1 if treated (here for treated individuals in the posttreatment period) and 0 otherwise; 3) the group level at which the treatment is received (here the individual); and 4) the time variable.

We obtain the same estimate of 2326.505. The default of the `xtdidregress` command is to obtain standard errors that cluster on the group variable, here `id`. The standard error of 644.6921 differs slightly from 644.7524, obtained earlier using the `regress` command, because of slightly different degrees-of-freedom correction.

The output also includes a table that gives the minimum and maximum of the first time that control individuals are observed (here time 1 for all 2,490 controls) and the minimum and maximum of the first time that treatment occurs (here time 2 for all 185 treated).

Further details on DID methods are presented in section 25.6. The `didregress` command adds individual-specific effects, and the `xtdidregress` command adds group-specific effects. The initial OLS regression of the current example excluded individual-specific effects yet led to the same estimate of the ATET because in the special case of a two-period binary treatment with no control variables, the individual-specific effects can be shown to be absorbed by inclusion of `dpost` and `dtreat`. The `nogteffects` option drops both group and time effects, so if this option is used, one needs to add in time effects as a control. The following yields an ATET of 2326.505.

```
. * xtdidregress command for difference in differences - no controls  
. xtdidregress (earns year) (dpostbydtreat), group(id) time(year) nogteffects  
(output omitted)
```

An important complication considered in section 25.6 is that treatment is often at a grouped level rather than the individual level, as was the case in the current application. For example, suppose some villages received treatment and some did not. In that case, even if individual-level data are available, standard errors of estimates need to be computed by clustering at the village level.

## 4.9 Additional resources

The `margins` command is very powerful, and many more examples are provided in the Stata documentation. The main linear regression commands `regress` and `regress postestimation` are covered in [R]. The regression decomposition commands `oaxaca` and `rego` are community contributed. The policy prediction example in section [4.2.4](#) and the DID example in section [4.8](#) are examples of treatment-effects estimators that are covered in detail in chapters 24 and 25.

## 4.10 Exercises

1. Fit the model in section [4.2](#) on levels, except use all observations rather than those with just positive expenditures, and report robust standard errors. Predict medical expenditures. Use `correlate` to obtain the correlation coefficient between the actual and fitted value, and show that, upon squaring, this equals  $R^2$ . Show that for the linear model `margins` with the `dydx(*)` option reproduces the OLS coefficients. Now, use `margins` with an appropriate option to obtain the average income elasticity of medical expenditures.
2. Consider the example of section [4.2.4](#) and the third method that computes predictions at `suppins==1` for all observations and at `suppins==0` and then compares the average predictions. The text did this following OLS regression in levels and obtained an estimated treatment effect of \$725. Repeat this example but with predictions from the log-linear model that correct for retransformation bias using Duan's method. You should obtain an estimate of \$2,048.
3. Fit the model in section [4.2](#) on levels, using the first 2,000 observations. Use these estimates to predict medical expenditures for the remaining 1,064 observations, and compare these with the actual values. Note that the model predicts very poorly in part because the data were ordered by `totexp`.
4. Reconsider the out-of-sample prediction analysis of section [4.3](#). Modify the DGP such that the regression intercept for the last 10 observations is 5, whereas it is fixed at 1 for the first 90 observations. Generate 10 out-of-sample predictions, and compare them with the case in which no such intercept shift occurs.
5. Repeat the analysis of the previous question but under the assumption that the intercept remains unchanged in the prediction period but the slope parameter increases or decreases by 20%. Compare the results with the no-change scenario.
6. Refer to the empirical example of section [4.3](#). Define two group-specific indicator variables, `d1` for the female group and `d2` for the male group. Sort the sample observations by group; females are in group 1. Using the full sample, run the earnings regression given in the

section; include  $d1$  and  $d2$  group indicators, but exclude the intercept term. Test the hypothesis that the coefficients of  $d1$  and  $d2$  are equal.

7. This is a continuation of the previous question. Replace regressors in the regression by new ones generated by multiplying the original regressors by  $d1$  and  $d2$ . The new regressors are group specific. Regress the earnings variable on the new generated group-specific regressors. Use the `test` command to test for the equality of the two sets of group-specific coefficients.



# **Chapter 5**

## **Simulation**

## 5.1 Introduction

Simulation by Monte Carlo experimentation is a useful and powerful methodology for investigating the properties of econometric estimators and tests. The power of the methodology derives from being able to define and control the statistical environment in which the investigator specifies the data-generating process (DGP) and generates data used in subsequent experiments.

Monte Carlo experiments can be used to verify that valid methods of statistical inference are being used. An obvious example is checking a new computer program or algorithm. Another example is investigating the robustness of an established estimation or test procedure to deviations from settings where the properties of the procedure are known.

Even when valid methods are used, they often rely on asymptotic results. We may want to check whether these provide a good approximation in samples of the size typically available to the investigators. Also, asymptotically equivalent procedures may have different properties in finite samples. Monte Carlo experiments enable finite-sample comparisons.

This chapter deals with the basic elements common to Monte Carlo experiments: computer generation of random numbers that mimic the theoretical properties of realizations of random variables; commands for repeated execution of a set of instructions; and machinery for saving, storing, and processing the simulation output generated in an experiment to obtain the summary measures that are used to evaluate the properties of the procedures under study. We provide a series of examples to illustrate various aspects of Monte Carlo analyses. Further examples are given throughout the book.

The chapter appears early in the book. Simulation is a powerful pedagogic tool for exposition and illustration of statistical concepts. At the simplest level, we can use pseudorandom samples to illustrate distributional features of artificial data. The goal of this chapter is to use simulation to study the distributional and moment properties of statistics in certain

idealized statistical environments. Another possible use of the Monte Carlo methodology is to check the correctness of computer code. Many applied studies use methods complex enough that it is easy to make mistakes. Often, these mistakes could be detected by an appropriate simulation exercise. We believe that simulation is greatly underutilized, even though Monte Carlo experimentation is relatively straightforward in Stata.

## 5.2 Pseudorandom-number generators

Suppose we want to use simulation to study the properties of the ordinary least-squares (OLS) estimator in the linear regression model with normal errors. Then, at the minimum, we need to make draws from a specified normal distribution. The literature on (pseudo)-random-number generation contains many methods of generating such sequences of numbers. When we use packaged functions, we usually do not need to know the details of the method. Yet the match between the theoretical and the sample properties of the draws does depend upon such details.

Stata has a suite of fast and easy-to-use random-number functions (generators). The suite includes the uniform, normal, binomial, gamma, and Poisson functions that we will use in this chapter, as well as several others that we do not use. The functions for generating pseudorandom numbers are summarized in `help random number functions`.

Underlying generation of any random variable is a uniform random-number generator. In a very major change, Stata 14 introduced a new uniform random-number generator that will lead to different random numbers being generated.

Thus, any results obtained using versions of Stata before version 14, and that used random numbers in any way, will change, unless appropriate commands are added to ensure random numbers are based on the original random-number generator.

### 5.2.1 Uniform random-number generation

The term “random-number generation” is an oxymoron. It is more accurate to use the term “pseudorandom numbers”. Pseudorandom-number generators use deterministic devices to produce long chains of numbers that mimic the realizations from some target distribution. For uniform random numbers, the target distribution is the uniform distribution from 0 to 1, for which any value between 0 and 1 is equally likely. Given such a sequence, methods

exist for mapping these into sequences of nonuniform draws from desired distributions such as the normal.

A standard simple generator for uniform draws uses the deterministic rule  $X_j = (kX_{j-1} + c) \bmod m$ ,  $j = 1, \dots, J$ , where the modulus operator  $a \bmod b$  forms the remainder when  $a$  is divided by  $b$  to produce a sequence of  $J$  integers between 0 and  $m - 1$ . Then  $R_j = X_j/m$  is a sequence of  $J$  numbers between 0 and 1. If computation is done using 32-bit integer arithmetic, then  $m = 2^{31} - 1$ , and the maximum periodicity is  $2^{31} - 1 \simeq 2.1 \times 10^9$ , but it is easy to select poor values of  $k$ ,  $c$ , and  $X_0$  so that the cycle repeats much more often than that. For computation using 64-bit integer arithmetic, the maximum periodicity is  $2^{61} - 1 \simeq 1.8 \times 10^{19}$ .

Before version 14, the Stata function `runiform()` used the 32-bit KISS generator. Version 14 replaced this with the 64-bit Mersenne Twister generator. Users of version 14 and later can continue to use the older KISS generator, if desired, by giving the command `set rng kiss32`.

The initial value for the cycle,  $X_0$ , is called the seed. The default is to have this set by Stata. For reproducibility of results, however, it is best to actually set the initial seed by using the `set seed` command or using the `seed()` option in, for example, `vce(bootstrap)`. Then, if the program is rerun at a later time or by a different researcher, the same results will be obtained; see exercise 1 in section [5.8](#).

To obtain and display one draw from the uniform, type

```
. * Single draw of a uniform number
. set seed 10101
. scalar u = runiform()
. display u
.30422325
```

This number is internally stored at much greater precision than the eight displayed digits.

The following code obtains 1,000 draws from the uniform distribution and then provides some details on these draws:

```

. * 1,000 draws of uniform numbers
. qui set obs 1000
. set seed 10101
. generate x = runiform()
. list x in 1/5, clean

      x
1.  .3042232
2.  .5540206
3.  .2794988
4.  .2006274
5.  .1246266

. summarize x

      Variable |       Obs        Mean   Std. dev.      Min      Max
                 | 1,000    .5031233    .2922443   .0005628   .9996441

```

The 1,000 draws have a mean of 0.503 and a standard deviation of 0.292, close to the theoretical values of 0.5 and  $\sqrt{1/12} = 0.289$ . A histogram, not given, has 10 equal-width bins with heights that range from 0.8 to 1.2, close to the theory of equal heights of 1.0.

The draws should be serially uncorrelated, despite a deterministic rule being used to generate the draws. To verify this, we create a time-identifier variable, *t*, equal to the observation number (*\_n*), and we use *tsset* to declare the data to be time series with time-identifier *t*. We could then use the *corrgram*, *ac*, and *pac* commands to test whether autocorrelations and partial autocorrelations are zero. We more simply use *pwcorr* to produce the first three autocorrelations, where *L2.x* is the *x* variable lagged twice and the *star(0.05)* option puts a star on correlations that are statistically significantly different from 0 at level 0.05.

```

. * First three autocorrelations for the uniform draws
. generate t = _n
. tsset t
Time variable: t, 1 to 1000
Delta: 1 unit
. pwcorr x L.x L2.x L3.x, star(0.05)

```

	x	L.x	L2.x	L3.x
x	1.0000			
L.x	0.0107	1.0000		
L2.x	-0.0004	0.0109	1.0000	
L3.x	-0.0210	0.0000	0.0108	1.0000

The autocorrelations are low, and none are statistically different from 0 at the 0.05 level. Uniform random-number generators used by packages such as Stata are, of course, subjected to much more stringent tests than these.

### 5.2.2 Draws from normal

For simulations of standard estimators such as OLS, nonlinear least squares, and instrumental variables (IV), all that is needed are draws from the uniform and normal distributions because normal errors are a natural starting point and the most common choices of distribution for generated regressors are normal and uniform.

The command for making draws from the standard normal has the following simple syntax:

```
generate varname = rnormal()
```

To make draws from  $N(m, s^2)$ , type the corresponding command

```
generate varname = rnormal(m,s)
```

Note that  $s > 0$  is the standard deviation. The arguments  $m$  and  $s$  can be numbers or variables.

Draws from the standard normal distribution also can be obtained as a transformation of draws from the uniform by using the inverse-probability transformation method explained in section [5.4.1](#), that is, by using

```
generate varname = invnormal(runiform())
```

This alternative method is computationally slower than using the `rnormal()` function.

The following code generates and summarizes three pseudorandom variables with 1,000 observations each. The pseudorandom variables have distributions uniform(0, 1), standard normal, and normal with a mean of 5 and a standard deviation of 2.

```
. * Normal and uniform and uniform draws
. clear
. qui set obs 1000
. set seed 10101                                // Set the seed
. generate uniform = runiform()                  // uniform(0,1)
. generate stnormal = rnormal()                  // N(0,1)
. generate norm5and2 = rnormal(5,2)
. tabstat uniform stnormal norm5and2, stat(mean sd skew kurt min max) col(stat)
```

Variable	Mean	SD	Skewness	Kurtosis	Min	Max
uniform	.5031233	.2922443	-.0264117	1.809826	.0005628	.9996441
stnormal	.0230095	1.023687	-.193405	3.12781	-4.119125	2.615001
norm5and2	5.090733	1.983719	-.1014361	2.986503	-1.140269	11.52456

The sample mean and other sample statistics are random variables; therefore, their values will, in general, differ from the true population values. As the number of observations grows, each sample statistic will converge to the population parameter because each sample statistic is a consistent estimator for the population parameter.

For `norm5and2`, the sample mean and standard deviation are very close to the theoretical values of 5 and 2. Output from `tabstat` gives a skewness statistic of – 0.101 and a kurtosis statistic of 2.987, close to 0 and 3, respectively.

For draws from the truncated normal, see section [5.4.4](#), and for draws from the multivariate normal, see section [5.4.5](#).

### 5.2.3 Draws from t, chi-squared, F, gamma, and beta

Stata's library of functions contains a number of generators that allow the user to draw directly from a number of common continuous distributions. The function formats are similar to that of the `rnormal()` function, and the arguments can be a number or a variable.

Let  $t(n)$  denote Student's  $t$  distribution with  $n$  degrees of freedom,  $\chi^2(m)$  denote the chi-squared distribution with  $m$  degrees of freedom, and  $F(h, n)$  denote the  $F$  distribution with  $h$  and  $n$  degrees of freedom. Draws from  $t(n)$  and  $\chi^2(h)$  can be made directly by using the `rt(df)` and `rchi2(df)` functions. We then generate  $F(h, n)$  draws by transformation because a function for drawing directly from the  $F$  distribution is not available.

The following example generates draws from  $t(10)$ ,  $\chi^2(10)$ , and  $F(10, 5)$ .

```
. * Student's t, chi-squared, and F draws with constant degrees of freedom
. clear
. qui set obs 2000
. set seed 10101
. generate xt = rt(10)                      // Result xt ~ t(10)
. generate xc = rchi2(10)                     // Result xc ~ chisquared(10)
. generate xfn = rchi2(10)/10                 // Result numerator of F(10,5)
. generate xfd = rchi2(5)/5                   // Result denominator of F(10,5)
. generate xf = xfn/xfd                      // Result xf ~ F(10,5)
. summarize xt xc xf
```

Variable	Obs	Mean	Std. dev.	Min	Max
xt	2,000	.0141702	1.144066	-5.294379	5.451671
xc	2,000	10.10297	4.585638	1.403877	32.81162
xf	2,000	1.687583	2.339381	.0818754	45.02815

The  $t(10)$  draws have a sample mean and a standard deviation close to the theoretical values of 0 and  $\sqrt{10/(10 - 2)} = 1.118$ ; the  $\chi^2(10)$  draws have a sample mean and a standard deviation close to the theoretical values of 10 and  $\sqrt{20} = 4.472$ ; and the  $F(10, 5)$  draws have a sample mean close to the theoretical value of  $5/(5 - 2) = 1.7$ . The sample standard deviation of 2.339 differs from the theoretical standard deviation of  $\sqrt{2 \times 5^2 \times 13/(10 \times 3^2 \times 1)} = 2.687$ . This is because of randomness, and a much larger number of draws eliminates this divergence.

Using `rbeta(a, b)`, we can draw from a two-parameter beta with the shape parameters  $a, b > 0$ , mean  $a/(a + b)$ , and variance  $ab/(a + b)^2(a + b + 1)$ . Using `rgamma(a, b)`, we can draw from a two-parameter gamma with the shape parameter  $a > 0$ , scale parameter  $b > 0$ , mean  $ab$ , and variance  $ab^2$ .

### 5.2.4 Draws from binomial, Poisson, and negative binomial

Stata functions also generate draws from some leading discrete distributions. Again, the arguments can be a number or a variable.

Let  $\text{Bin}(n, p)$  denote the binomial distribution with positive integer  $n$  trials ( $n$ ) and success probability  $p$ ,  $0 < p < 1$ , and let  $\text{Poisson}(m)$  denote the Poisson distribution with the mean or rate parameter  $m$ . The `rbinomial(n, p)` function generates random draws from the binomial distribution, and the `rpoisson(m)` function makes draws from the Poisson distribution. Additionally, we use the `runiformint(a, b)` function, which generates equally likely integer values between integer  $a$  and integer  $b$ .

We demonstrate these functions with an argument that is a variable so that the parameters differ across draws.

#### Independent (but not identically distributed) draws from the binomial

As illustration, we consider draws from the binomial distribution, when both the probability  $p$  and the number of trials  $n$  may vary over  $i$ .

```
. * Discrete random variables: Binomial draws with n and p varying over trials
. set seed 10101
. generate p1 = runiform()                      // Here p1~uniform(0,1)
. generate trials = runiformint(1,10)           // Here # trials varies btwn 1 & 10
. generate xbin = rbinomial(trials,p1)          // Draws from binomial(n,p1)
. summarize p1 trials xbin
```

Variable	Obs	Mean	Std. dev.	Min	Max
p1	2,000	.4974725	.2909053	.0005628	.9996441
	2,000	5.497	2.850097	1	10
	2,000	2.7325	2.479116	0	10

The DGP setup implies that the number of trials  $n$  is a random variable with an expected value of 5.5 and that the probability  $p$  is a random variable with an expected value of 0.5. Thus, we expect that `xbin` has a mean of  $5.5 \times 0.5 = 2.75$ , and this is approximately the case here.

### Independent (but not identically distributed) draws from Poisson

For simulating a Poisson regression DGP, denoted  $y \sim \text{Poisson}(\mu)$ , we need to make draws that are independent but not identically distributed, with the mean  $\mu$  varying across draws because of regressors.

We do so in two ways. First, let  $\mu_i$  equal `xb=4+2*x` with `x=runiform()`. Then  $4 < \mu_i < 6$ . Second, let  $\mu_i$  equal `xb` times `xg` where `xg=rgamma(1, 1)`, which yields a draw from the gamma distribution with a mean of  $1 \times 1 = 1$  and a variance of  $1 \times 1^2 = 1$ . Then  $\mu_i > 0$ . In both cases, the setup can be shown to be such that the ultimate draw has a mean of 5, but the variance differs from 5 for the independent and identically distributed (i.i.d.) Poisson because in neither case are the draws from an identical distribution. We obtain

```
. * Discrete random variables: Poisson and negative binomial draws i.i.d.
. set seed 10101
. generate xb= 4 + 2*runiform()
. generate xg = rgamma(1,1)                                // Draw from gamma;E(v)=1
. generate xbh = xb*xg                                     // Apply multiplicative heterogeneity
. generate xp = rpoisson(5)                                 // Result xp ~ Poisson(5)
. generate xp1 = rpoisson(xb)                               // Result xp1 ~ Poisson(xb)
. generate xp2 = rpoisson(xbh)                             // Result xp2 ~ NB(xb)
. summarize xg xb xp xp1 xp2
```

Variable	Obs	Mean	Std. dev.	Min	Max
xg	2,000	1.000795	1.011366	.0016444	10.49622
xb	2,000	4.994945	.5818107	4.001126	5.999288
xp	2,000	5.001	2.210077	0	16
xp1	2,000	4.977	2.353559	0	14
xp2	2,000	4.928	5.509991	0	50

The `xb` variable lies between 4 and 6, as expected, and the `xg` gamma variable has a mean and variance close to 1, as expected. For a benchmark comparison, we make draws of `xp` from  $\text{Poisson}(5)$ , which has a sample

mean close to 5 and a sample standard deviation close to  $\sqrt{5} = 2.236$ . Both `xp1` and `xp2` have means close to 5. In the case of `xp2`, the model has the multiplicative unobserved heterogeneity term `xg`, which is itself drawn from a gamma distribution with shape and scale parameter both set to 1.

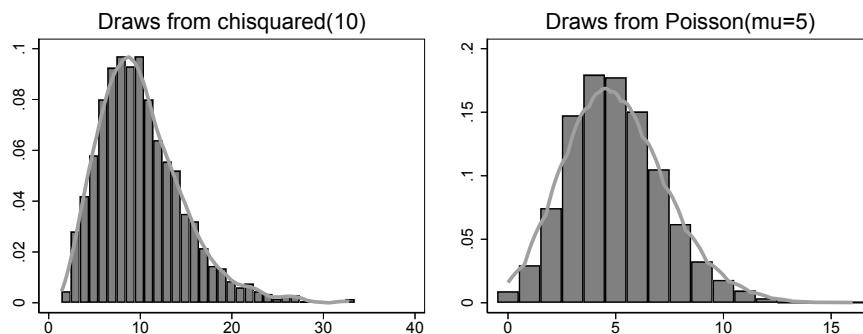
Introducing this type of heterogeneity means that `xp2` is drawn from a distribution with the same mean as that of `xp1`, but the variance of the distribution is larger. More specifically,  $\text{Var}(xp2|xb) = xb*(1+xb)$ , using results in section 20.2.2, leading to the much larger standard deviation for `xp2`.

The second example makes a draw from the Poisson–gamma mixture, yielding the negative binomial distribution. The `rnbnomial()` function draws from a different parameterization of the negative binomial distribution. Thus, we draw from the Poisson–gamma mixture here and in section 20.2.

### Histograms and density plots

For a visual depiction, it is often useful to plot a histogram or kernel density estimate of the generated random numbers. Here we do this for the draws `xc` from  $\chi^2(10)$  and `xp` from  $\text{Poisson}(5)$ . The results are shown in figure 5.1.

```
. * Example of histogram and kernel density plus graph combine
. qui twoway (histogram xc, width(1)) (kdensity xc, lwidth(thick)),
> legend(off) title("Draws from chisquared(10)") saving(graph1.gph, replace)
. qui twoway (histogram xp, discrete) (kdensity xp, lwidth(thick) w(1)),
> legend(off) title("Draws from Poisson(mu=5)") saving(graph2.gph, replace)
. graph combine graph1.gph graph2.gph, iscale(1.2) ysize(2.5) xsize(6.0)
```



**Figure 5.1.**  $\chi^2(10)$  and Poisson(5) draws

## 5.3 Distribution of the sample mean

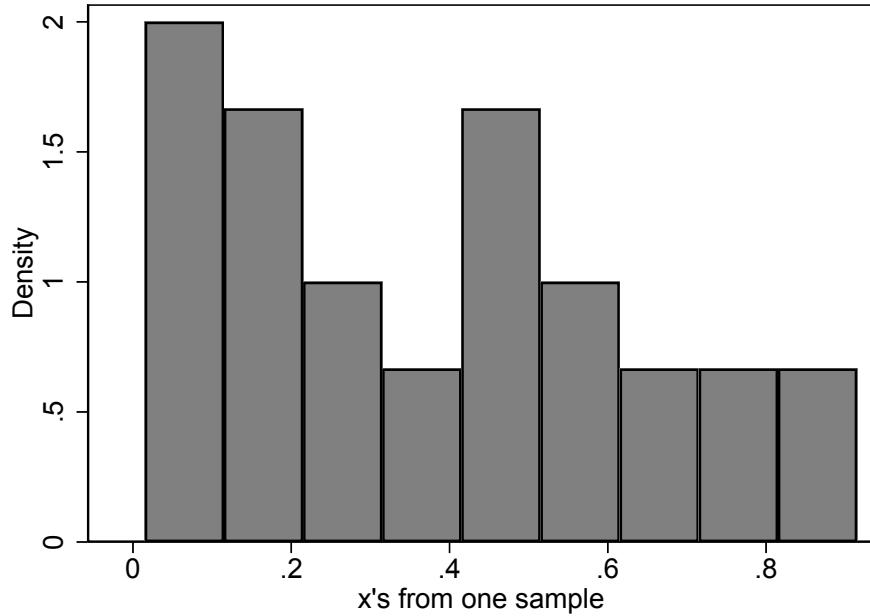
As an introductory example of simulation, we demonstrate the central limit theorem result,  $\sqrt{N}(\bar{x}_N - \mu)/\sigma \rightarrow N(0, 1)$ ; that is, the sample mean is approximately normally distributed as  $N \rightarrow \infty$ . We consider a random variable that has the uniform distribution and a sample size of 30.

We begin by drawing a single sample of size 30 of the random variable  $X$  that is uniformly distributed on  $(0, 1)$ , using the `runiform()` random-number function. To ensure the same results are obtained in future runs of the same code or on other machines, we use `set seed`. We have

```
. * Draw 1 sample of size 30 from uniform distribution
. clear all
. qui set obs 30
. set seed 10101
. generate x = runiform()
```

To see the results, we use `summarize` and `histogram`. We have

```
. * Summarize x and produce a histogram
. summarize x
      Variable |       Obs        Mean    Std. dev.      Min      Max
                 |   30     .380269     .2706183     .014999     .886691
. histogram x, width(0.1) xtitle("x's from one sample") scale(1.2)
(bin=9, start=.01499897, width=.1)
```



**Figure 5.2.** Histogram for one sample of size 30

The summary statistics show that 30 observations were generated and that for this sample  $\bar{x} = 0.380$ .

The histogram for this single sample of 30 uniform draws, given in figure 5.2, looks nothing like the bell-shaped curve of a normal because we are sampling from the uniform distribution. For very large samples, this histogram approaches a horizontal line with a density value of 1.

To obtain the distribution of the sample mean by simulation, we redo the preceding 10,000 times, obtaining 10,000 samples of size 30 and 10,000 sample means  $\bar{x}$ . These 10,000 sample means are draws from the distribution of the sample-mean estimator. By the central limit theorem, the distribution of the sample-mean estimator has approximately a normal distribution. Because the mean of a uniform(0, 1) distribution is 0.5, the mean of the distribution of the sample-mean estimator is 0.5. Because the standard deviation of a uniform(0, 1) distribution is  $\sqrt{1/12}$  and each of the 10,000 samples is of size 30, the standard deviation of the distribution of the sample-mean estimator is  $\sqrt{(1/12)/30} = 0.0527$ .

### 5.3.1 Stata program

A mechanism for repeating the same statistical procedure 10,000 times is to write a program (see appendix [A.2](#) for more details) that does the procedure once and use the `simulate` prefix to run the program 10,000 times.

We name the program `onesample` and define it to be r-class, meaning that the ultimate result, the sample mean for one sample, is returned in `r()`. Because we name this result `meanforonesample`, it will be returned in `r(meanforonesample)`. The program has no inputs, so there is no need for program arguments. The program drops any existing data on variable `x`, sets the sample size to 30, draws 30 uniform variates, and obtains the sample mean with `summarize`. The `summarize` command is itself an r-class command that stores the sample mean in `r(mean)`; see section [1.6.1](#). The last line of the program returns `r(mean)` as the result `meanforonesample`.

The program is

```
. * Program to draw 1 sample of size 30 from uniform and return sample mean
. program onesample, rclass
  1. drop _all
  2. qui set obs 30
  3. generate x = runiform()
  4. summarize x
  5. return scalar meanforonesample = r(mean)
  6. end
```

To check the program, we run it once, using the same seed as earlier. We obtain

```
. * Run program onesample once as a check
. set seed 10101
. onesample
      Variable |       Obs        Mean    Std. dev.       Min       Max
                 x |       30     .380269     .2706183     .014999     .886691
. return list
scalars:
r(meanforonesample) =  .3802689793519676
```

The results for one sample are exactly the same as those given earlier.

### 5.3.2 The `simulate` prefix

The `simulate` prefix runs a specified command # times, where the user specifies #. The basic syntax is

```
simulate [exp_list], reps(#) [options]: command
```

where *command* is the name of the command, often a user-written program, and # is the number of simulations or replications. The quantities to be calculated and stored from *command* are given in *exp\_list*. We provide additional details on `simulate` in section [5.6.1](#).

After `simulate` is run, the Stata dataset currently in memory is replaced by a dataset that has # observations, with a separate variable for each of the quantities given in *exp\_list*.

### 5.3.3 Central limit theorem simulation

The `simulate` prefix can be used to run the `onesample` program 10,000 times, yielding 10,000 sample means from samples of size 30 of uniform variates. We additionally used options that set the seed and suppress the output of a dot for each of the 10,000 simulations. We have

```
. * Run program onesample 10,000 times to get 10,000 sample means  
. simulate xbar = r(meanforonesample), seed(10101) reps(10000) nodots: onesample  
    Command: onesample  
          xbar: r(meanforonesample)
```

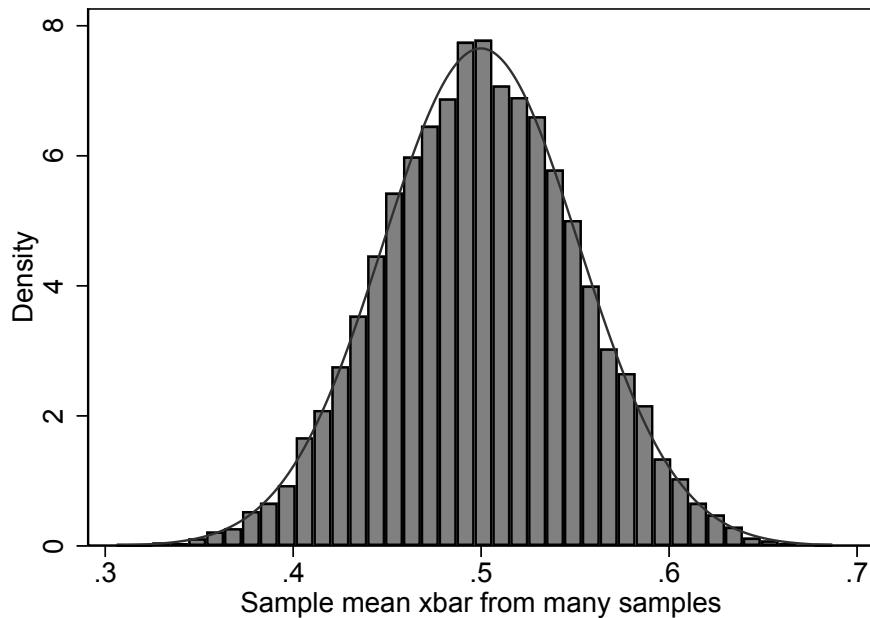
The result from each sample, `r(meanforonesample)`, is stored as the variable `xbar`.

The `simulate` prefix overwrites any existing data with a dataset of 10,000 “observations” on  $\bar{x}$ . We summarize these values, expecting them to have a mean close to 0.5 and a standard deviation close to 0.0527. We also present a histogram overlaid by a normal density curve with a mean and standard deviation, which are those of the 10,000 values of  $\bar{x}$ . We have

```
. * Summarize the 10,000 sample means and draw histogram  
. summarize xbar
```

Variable	Obs	Mean	Std. dev.	Min	Max
xbar	10,000	.499825	.0521404	.306084	.6868161

```
. histogram xbar, normal xtitle("Sample mean xbar from many samples") scale(1.2)
(bin=40, start=.30608404, width=.0095183)
```



**Figure 5.3.** Histogram of the 10,000 sample means, each from a sample of size 30

The histogram given in figure 5.3 is very close to the bell-shaped curve of the normal.

There are several possible variations on this example. Different distributions for `x` can be used with different random-number functions in the `generate` command for `x`. As sample size (`set obs`) and number of simulations (`reps`) increases, the results become closer to a normal distribution.

### 5.3.4 The `postfile` command

In this book, we generally use `simulate` to perform simulations. An alternative method is to use a looping command, such as `forvalues`, and, within each iteration of the loop, use `post` to write (or `post`) key results to a file that is declared in the `postfile` command. After the loop ends, we then analyze the data in the posted file.

The `postfile` command has the following basic syntax,

```
postfile postname newvarlist using filename [ , every(#) replace ]
```

where *postname* is an internal filename, *newvarlist* contains the names of the variables to be put in the dataset, and *filename* is the external filename.

The `post` *postname* (*exp1*) (*exp2*) ... command is used to write *exp1*, *exp2*, ... to the file. Each expression needs to be enclosed in parentheses.

The `postclose` *postname* command ends the posting of observations.

The `postfile` command offers more flexibility than `simulate` and, unlike `simulate`, does not lead to the dataset in memory being overwritten. For the examples in this book, `simulate` is adequate.

### 5.3.5 Alternative central limit theorem simulation

We illustrate the use of `postfile` for the central limit theorem example. We have

```
. * Simulation using postfile
. set seed 10101
. postfile sim_mem xmean using simresults, replace
(file simresults.dta not found)
. forvalues i = 1/10000 {
  2. drop _all
  3. qui set obs 30
  4. tempvar x
  5. generate `x' = runiform()
  6. qui summarize `x'
  7. post sim_mem (r(mean))
  8. }
. postclose sim_mem
```

The `postfile` command declares the memory object in which the results are stored, the names of variables in the results dataset, and the name of the results dataset file. In this example, the memory object will be named `sim_mem`, `xmean` will be the only variable in the results dataset file, and `simresults.dta` will be the results dataset file. (The `replace` option causes any existing `simresults.dta` to be replaced.) The `forvalues` loop (see

section 1.8) is used to perform 10,000 repetitions. At each repetition, the sample mean, result `r(mean)`, is posted and will be included as an observation in the new `xmean` variable in `simresults.dta`.

To see the results, we need to open `simresults.dta` and summarize.

```
. * See the results stored in simresults  
. use simresults, clear  
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
xmean	10,000	.499825	.0521404	.306084	.6868161

The results are identical to those in section 5.3.3 with `simulate` due to using the same seed and same sequence of evaluation of random-number functions.

The `simulate` prefix suppresses all output within the simulations. This is not the case for the `forvalues` loop, so the `quietly` prefix was used in two places in the code above. It can be more convenient to instead apply the `quietly` prefix to all commands in the entire `forvalues` loop.

## 5.4 Pseudorandom-number generators: Further details

In this section, we present further details on random-number generation that explain the methods used in section [5.2](#) and are useful for making draws from additional distributions.

Commonly used methods for generating pseudorandom samples include inverse-probability transforms, direct transformations, accept–reject methods, mixing and compounding, and Markov chains. In what follows, we emphasize application and refer the interested reader to [Cameron and Trivedi \(2005\)](#), chap. 12 or numerous other texts for additional details.

### 5.4.1 Inverse-probability transformation

Let  $F(x) = \Pr(X \leq x)$  denote the cumulative distribution function (c.d.f.) of a random variable  $x$ . Given a draw of a uniform variate  $r$ ,  $0 \leq r \leq 1$ , the inverse transformation  $x = F^{-1}(r)$  gives a unique value of  $x$  because  $F(x)$  is nondecreasing in  $x$ . If  $r$  approximates well a random draw from the uniform, then  $x = F^{-1}(r)$  will approximate well a random draw from  $F(x)$

A leading application is to the standard normal distribution. Then the inverse of the c.d.f.,

$$F(x) = \Phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz$$

has no closed-form solution, and there is consequently no analytical expression for  $\Phi^{-1}(x)$ . Nonetheless, the inverse-transformation method is easy to implement because numerical analysis provides functions that calculate a very good approximation to  $\Phi^{-1}(x)$ . In Stata, the function is `invnormal()`. Combining the two steps of drawing a random uniform variate and evaluating the inverse c.d.f., we have

```

. * Inverse-probability transformation example: Standard normal
. clear all
. qui set obs 2000
. set seed 10101
. generate xstn = invnormal(runiform())

```

This was presented in section [5.2.2](#) but is now superseded by the faster `rnormal()` function.

As another application, consider drawing from the unit exponential, with c.d.f.  $F(x) = 1 - e^{-x}$ . Solving  $r = 1 - e^{-x}$  yields  $x = -\ln(1 - r)$ . If the uniform draw is, say, 0.640, then  $x = -\ln(1 - 0.640) = 1.022$ . With continuous monotonically increasing c.d.f., the inverse transformation yields a unique value of  $x$ , given  $r$ . The Stata code for generating a draw from the unit exponential illustrates the method:

```

. * Inverse-probability transformation example: Unit exponential
. generate xue = -ln(1-runiform())

```

For discrete random variables, the c.d.f. is a step function. Then the inverse is not unique, but it can be uniquely determined by a convention for choosing a value on the flat portion of the c.d.f., for example, the left limit of the segment.

In the simplest case, we consider a Bernoulli random variable taking a value of 1 with a probability of  $p$  and a value of 0 with a probability of  $1 - p$ . Then we take a uniform draw,  $u$ , and set  $y = 1$  if  $u \leq p$  and  $y = 0$  if  $u > p$ . Thus, if  $p = 0.6$ , we obtain the following:

```

. * Inverse-probability transformation example: Bernoulli (p = 0.6)
. generate xbernoulli = runiform() > 0.6 // Bernoulli(0.6)
. summarize xstn xue xbernoulli

```

Variable	Obs	Mean	Std. dev.	Min	Max
xstn	2,000	-.0114795	1.012011	-3.257087	3.384994
xue	2,000	.9920891	.9782134	.0000276	6.411174
xbernoulli	2,000	.3885	.4875311	0	1

This code uses a logical operator that sets  $y = 1$  if the condition is met and  $y = 0$  otherwise; see section [2.4.7](#).

A more complicated discrete example is the Poisson distribution because then the random variable can potentially take an infinite number of values. The method is to sequentially calculate the c.d.f.  $\Pr(Y \leq k)$  for  $k = 0, 1, 2, \dots$ . Then, stop when the first  $\Pr(Y \leq k) > u$ , where  $u$  is the uniform draw, and set  $y = k$ . For example, consider the Poisson with a mean of 2 and a uniform draw of 0.701. We first calculate  $\Pr(y \leq 0) = 0.135 < u$ , then calculate  $\Pr(y \leq 1) = 0.406 < u$ , then calculate  $\Pr(y \leq 2) = 0.677 < u$ , and finally calculate  $\Pr(y \leq 3) = 0.857$ . This last calculation exceeds the uniform draw of 0.701, so stop and set  $y = 3$ .  $\Pr(Y \leq k)$  is computed by using the recursion  $\Pr(Y \leq k) = \Pr(Y \leq k - 1) + \Pr(Y = k)$ .

### 5.4.2 Direct transformation

Suppose we want to make draws from the random variable  $Y$ , and from probability theory, it is known that  $Y$  is a transformation of the random variable  $X$ , say,  $Y = g(X)$ .

In this situation, the direct transformation method obtains draws of  $Y$  by drawing  $X$  and then applying the transformation  $g(\cdot)$ . The method is clearly attractive when it is easy to draw  $X$  and evaluate  $g(\cdot)$ .

Direct transformation is particularly easy to illustrate for well-known transforms of a standard normally distributed random variable. A  $\chi^2(1)$  draw can be obtained as the square of a draw from the standard normal; a  $\chi^2(m)$  draw is the sum of  $m$  independent draws from  $\chi^2(1)$ ; an  $F(m_1, m_2)$  draw is  $(v_1/m_1)/(v_2/m_2)$ , where  $v_1$  and  $v_2$  are independent draws from  $\chi^2(m_1)$  and  $\chi^2(m_2)$ ; and a  $t(m)$  draw is  $u/\sqrt{v/m}$ , where  $u$  and  $v$  are independent draws from  $N(0, 1)$  and  $\chi^2(m)$ .

### 5.4.3 Other methods

In some cases, a distribution can be obtained as a mixture of distributions. A leading example is the negative binomial, which can be obtained as a Poisson–gamma mixture (see section [5.2.4](#)). Specifically, if  $y|\lambda$  is  $\text{Poisson}(\lambda)$  and  $\lambda|\mu, \alpha$  is gamma with a mean of  $\mu$  and a variance of  $\alpha\mu$ , then  $y|\mu, \alpha$  is a negative binomial distributed with a mean of  $\mu$  and a variance of  $\mu + \alpha\mu^2$ . This implies that we can draw from the negative

binomial by using a two-step method in which we first draw (say,  $\nu$ ) from the gamma distribution with a mean equal to 1 and then, conditional on  $\nu$ , draw from  $\text{Poisson}(\mu\nu)$ . This example, using mixing, is used again in chapter 20.

More advanced methods include accept–reject algorithms and importance sampling. Many of Stata’s pseudorandom-number generators use accept–reject algorithms. Type `help random number functions` for more information on the methods used by Stata.

#### 5.4.4 Draws from truncated normal

In simulation-based estimation for latent normal models with censoring or selection, it is often necessary to generate draws from a truncated normal distribution. The inverse-probability transformation can be extended to obtain draws in this case.

Consider making draws from a truncated normal. Then  $X \sim TN_{(a,b)}(\mu, \sigma^2)$ , where without truncation  $X \sim N(\mu, \sigma^2)$ . With truncation, realizations of  $X$  are restricted to lie between left truncation point  $a$  and right truncation point  $b$ .

For simplicity, first consider the standard normal case ( $\mu = 0, \sigma = 1$ ), and let  $Z \sim N(0, 1)$ . Given the draw  $u$  from the uniform distribution,  $x$  is defined by the solution of the inverse-probability transformation equation

$$u = \frac{\Pr(a \leq Z \leq x)}{\Pr(a \leq Z \leq b)} = \frac{\Phi(x) - \Phi(a)}{\Phi(b) - \Phi(a)}$$

Rearranging,  $\Phi(x) = \Phi(a) + \{\Phi(b) - \Phi(a)\}u$  so that solving for  $x$ , we obtain

$$x = \Phi^{-1}[\Phi(a) + \{\Phi(b) - \Phi(a)\}u]$$

To extend this to the general case, note that if  $Z \sim N(\mu, \sigma^2)$  then  $Z^* = (Z - \mu)/\sigma \sim N(0, 1)$ , and the truncation points for  $Z^*$ , rather than  $Z$ , are  $a^* = (a - \mu)/\sigma$  and  $b^* = (b - \mu)/\sigma$ . Then,

$$x = \mu + \sigma\Phi^{-1}[\Phi(a^*) + \{\Phi(b^*) - \Phi(a^*)\} u]$$

As an example, we consider draws from  $N(5, 4^2)$  for a random variable truncated to the range  $[0, 12]$ .

```
. * Draws from truncated normal x ~ N(mu,sigma^2) in [a,b]
. qui set obs 2000
. set seed 10101
. scalar a = 0                                // Lower truncation point
. scalar b = 12                               // Upper truncation point
. scalar mu = 5                                // Mean
. scalar sigma = 4                            // Standard deviation
. generate u = runiform()
. generate w=normal((a-mu)/sigma)+u*(normal((b-mu)/sigma)-normal((a-mu)/sigma))
. generate xtrunc = mu + sigma*invnormal(w)
. summarize xtrunc
```

Variable	Obs	Mean	Std. dev.	Min	Max
xtrunc	2,000	5.421001	2.973021	.0105123	11.98595

Here there is more truncation from below because  $a$  is  $1.25\sigma$  from  $\mu$ , whereas  $b$  is  $1.75\sigma$  from  $\mu$ , so we expect the truncated mean to exceed the untruncated mean. Accordingly, the sample mean is 5.421 compared with the untruncated mean of 5. Truncation reduces the range and, for most but not all distributions, will reduce the variability. The sample standard deviation of 2.973 is less than the untruncated standard deviation of 4.

An alternative way to draw  $X \sim TN_{(a,b)}(\mu, \sigma^2)$  is to keep drawing from untruncated  $N(\mu, \sigma^2)$  until the realization lies in  $(a, b)$ . This method will be very inefficient if, for example,  $(a, b) = (-0.01, 0.01)$ . A Poisson example is given in section 20.4.

#### 5.4.5 Draws from multivariate normal

Making draws from multivariate distributions is generally more complicated. The method depends on the specific case under consideration, and inverse-transformation methods and transformation methods that work in the univariate case may no longer apply.

However, making draws from the multivariate normal is relatively straightforward because, unlike most other distributions, linear combinations of normals are also normal.

### Direct draws from multivariate normal

The `drawnorm` command generates draws from  $N(\mu, \Sigma)$  for the user-specified vector  $\mu$  and matrix  $\Sigma$ . For example, consider making 200 draws from a standard bivariate normal distribution with means of 10 and 20, variances of 4 and 9, and a correlation of 0.5 (so the covariance is 3).

```
. * Bivariate normal example: Means 10, 20; variances 4, 9; and correlation 0.5
. clear
. qui set obs 1000
. set seed 10101
. matrix MU = (10,20)                                // MU is 2 x 1
. scalar sig12 = 0.5*sqrt(4*9)
. matrix SIGMA = (4, sig12 \ sig12, 9)    // SIGMA is 2 x 2
. drawnorm y1 y2, means(MU) cov(SIGMA)
. summarize y1 y2

Variable |       Obs        Mean      Std. dev.       Min       Max
y1 |     1,000    10.00834     2.026031    1.76175   16.18743
y2 |     1,000    20.11815     3.012645   10.63777   30.54572
. correlate y1 y2
(obs=1,000)

y1 |       y1        y2
y1 |     1.0000
y2 |     0.4990     1.0000
```

The sample means are close to 10 and 20, and the standard deviations are close to  $\sqrt{4} = 2$  and  $\sqrt{9} = 3$ . The sample correlation of 0.499 differs very slightly from 0.50.

### Transformation using Cholesky decomposition

The method uses the result that if  $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ , then  $\mathbf{x} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z} \sim N(\boldsymbol{\mu}, \mathbf{LL}')$ . It is easy to draw  $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$  because  $\mathbf{z}$  is just a column vector of univariate normal draws. The transformation method to make draws of  $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$  evaluates  $\mathbf{x} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}$ , where the matrix  $\mathbf{L}$  satisfies  $\mathbf{LL}' = \boldsymbol{\Sigma}$ . More than one matrix  $\mathbf{L}$  satisfies  $\mathbf{LL}' = \boldsymbol{\Sigma}$ , the matrix analog of the square root of  $\boldsymbol{\Sigma}$ . Standard practice is to use the Cholesky decomposition that restricts  $\mathbf{L}$  to be a lower triangular matrix. Specifically, for the trivariate normal distribution, let  $\text{Var}(\mathbf{x}) = \boldsymbol{\Sigma} = \mathbf{L}\mathbf{z}\mathbf{z}'\mathbf{L}'$ , where  $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I}_3)$  and

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

Then the following transformations of  $\mathbf{z}' = (z_1 \ z_2 \ z_3)$  yield the desired multivariate normal vector  $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ :

$$\begin{aligned} x_1 &= \mu_1 + l_{11}z_1 \\ x_2 &= \mu_2 + l_{21}z_1 + l_{22}z_2 \\ x_3 &= \mu_3 + l_{31}z_1 + l_{32}z_2 + l_{33}z_3 \end{aligned}$$

#### 5.4.6 Draws using Markov chain Monte Carlo method

In some cases, making direct draws from a target joint (multivariate) distribution is difficult, so the objective must be achieved in a different way. However, if it is also possible to make draws from the distribution of a subset, conditional on the rest, then one can create a Markov chain of draws. If one recursively makes draws from the conditional distribution and if a sufficiently long chain is constructed, then the distribution of the draws will, under some conditions, converge to the distribution of independent draws from the stationary joint distribution. This so-called Markov chain Monte Carlo method is now standard in modern Bayesian inference, presented in chapters 29 and 30.

To be concrete, let  $\mathbf{Y} = (Y_1, Y_2)$  have a bivariate density of  $f(\mathbf{Y}) = f(Y_1, Y_2)$ , and suppose that the two conditional densities  $f(Y_1|Y_2)$  and  $f(Y_2|Y_1)$  are known and that it is possible to make draws from these. Then it can be shown that alternating sequential draws from  $f(Y_1|Y_2)$  and  $f(Y_2|Y_1)$  converge in the limit to draws from  $f(Y_1, Y_2)$ , even though in general  $f(Y_1, Y_2) \neq f(Y_1|Y_2)f(Y_2|Y_1)$  [recall that  $f(Y_1, Y_2) = f(Y_1|Y_2)f(Y_2)$ ]. The repeated recursive sampling from  $f(Y_1|Y_2)$  and  $f(Y_2|Y_1)$  is called the Gibbs sampler.

We illustrate the Markov chain Monte Carlo approach by making draws from a bivariate normal distribution,  $f(Y_1, Y_2)$ . Of course, when we use the `drawnorm` command, it is quite straightforward to draw samples from the bivariate normal. So the application presented is illustrative rather than practical. The relative simplicity of this method comes from the fact that the conditional distributions  $f(Y_1|Y_2)$  and  $f(Y_2|Y_1)$  derived from a bivariate normal are also normal.

We draw bivariate normal data with means of 0, variances of 1, and a correlation of  $\rho = 0.9$ . Then  $Y_1|Y_2 = y_2 \sim N\{\rho y_2, (1 - \rho^2)\}$  and  $Y_2|Y_1 = y_1 \sim N\{\rho y_1, (1 - \rho^2)\}$ . Implementation requires looping that is much easier using matrix programming language commands. The following Mata code implements this algorithm by using commands explained in appendix [B.2](#).

```
. * MCMC example: Gibbs for bivariate normal mu's=0 v's=1 corr=rho=0.9
. set seed 10101
. clear all
. set obs 1000
Number of observations (_N) was 0, now 1,000.
. generate double y1 =
(1,000 missing values generated)
. generate double y2 =
(1,000 missing values generated)
```

```

. mata:
----- mata (type end to exit) -----
: s0 = 10000          // Burn-in for the Gibbs sampler (to be discarded)
: s1 = 1000           // Actual draws used from the Gibbs sampler
: y1 = J(s0+s1,1,0)   // Initialize y1
: y2 = J(s0+s1,1,0)   // Initialize y2
: rho = 0.90          // Correlation parameter
: for(i=2; i<=s0+s1; i++) {
>     y1[i,1] = ((1-rho^2)^0.5)*(rnormal(1, 1, 0, 1)) + rho*y2[i-1,1]
>     y2[i,1] = ((1-rho^2)^0.5)*(rnormal(1, 1, 0, 1)) + rho*y1[i,1]
> }
: y = y1,y2
: y = y[|(s0+1),1 \ (s0+s1),.|] // Drop the burn-ins
: mean(y)                // Means of y1, y2
               1             2
1  .0629698832  .0577031161

: variance(y)           // Variance matrix of y1, y2
[symmetric]
               1             2
1  .8735151552
2  .7789225722  .8794946316

: correlation(y)        // Correlation matrix of y1, y2
[symmetric]
               1             2
1  1
2  .8886739931  1

: end
-----
```

Many draws may be needed before the chain converges. Here we assume that 10,000 draws are sufficient, and we discard the first 10,000 draws; the remaining 1,000 draws are kept. In a real application, one should run careful checks to ensure that the chain has indeed converged to the desired bivariate normal. For the example here, the sample means of  $Y_1$  and  $Y_2$  are 0.0630 and 0.0577, differing quite a bit from 0. Similarly, the sample variances of 0.874 and 0.879 differ from 1. The correlation of 0.889 is close to the desired 0.9. A longer Markov chain or longer burn-in may be needed to generate numbers with desired properties for this example with relatively high  $\rho$ .

Even given convergence of the Markov chain, the sequential draws of any random variable will be correlated. The output below shows that for the example here, the first-order correlation of sequential draws of  $y_2$  is 0.786.

```
. mata:  
: y2 = y[|2,2 \ s1,2|] mata (type end to exit)  
: y2lag1 = y[|1,2 \ (s1-1),2|]  
: y2andlag1 = y2,y2lag1  
: correlation(y2andlag1,1)      // Correlation between y2 and y2 lag 1  
[symmetric]  
          1           2  
1    1  
2 .7864381826           1  
: end
```

## 5.5 Computing integrals

Some estimation problems may involve definite or indefinite integrals. In such cases, the integral may be numerically calculated.

### 5.5.1 Quadrature

For one-dimensional integrals of the form  $\int_a^b f(y)dy$ , where possibly  $a = -\infty$ ,  $b = \infty$ , or both, Gaussian quadrature is the standard method. This approximates the integral by a weighted sum of  $m$  terms, where a larger  $m$  gives a better approximation and often even  $m = 20$  can give a good approximation. The formulas for the weights are quite complicated but are given in standard numerical analysis books.

Gauss–Hermite quadrature applies to the special case of unbounded integrals of form  $\int_{-\infty}^{\infty} e^{-y^2} r(y)dy$ . Gauss–Hermite quadrature approximates this integral by the sum  $\sum_{j=1}^m w_j r(y_j)$ , where the user provides the number of evaluation points  $m$ , while the weights  $w_j$  and evaluation points  $y_j$  are determined by the first  $m$  orthonormal Hermite polynomials.

One-dimensional integrals often appear in regression models with a random intercept or random effect. In many nonlinear models, this random effect does not integrate out analytically. Most often, the random effect is normal so that integration is over  $(-\infty, \infty)$ , and Gauss–Hermite quadrature is used. A leading example is the random-effects estimator for nonlinear panel models fit using various `xt` commands.

In higher dimensions, Gauss–Hermite quadrature does not always provide an adequate approximation, and adaptive Gauss–Hermite quadrature may provide better approximation. The Stata `me` and `gsem` commands for nonlinear models offer as possible methods mean-variance adaptive Gauss–Hermite quadrature, mode-curvature adaptive Gauss–Hermite quadrature, and nonadaptive Gauss–Hermite quadrature. The quadrature methods use a Cholesky decomposition to reduce a multidimensional problem to a series of one-dimensional Gauss–Hermite quadratures. See [ME] **meglm** for a detailed discussion.

For normal integrals, an alternative is to use a Laplacian approximation. For simplicity, consider the scalar case. Suppose  $f(y)$  has a maximum at  $y_0$ . Then  $f(y) \simeq f''(y_0)(y - y_0)^2/2$  by a second-order Taylor expansion. The integral  $\int_{-\infty}^{\infty} e^{f(y)} dy$  is then approximated by  $\int_{-\infty}^{\infty} e^{f''(y_0)(y-y_0)^2/2} dy = \sqrt{2\pi/|f''(y_0)|} e^{f(y_0)}$ . The `me` and `gsem` commands provide Laplacian approximation as an option; it is not as accurate as Gauss–Hermite quadrature but is faster.

### 5.5.2 Monte Carlo integration

Suppose the integral is of the form

$$E \{ h(Y) \} = \int_a^b h(y) g(y) dy$$

where  $g(y)$  is a density function. This can be estimated by the direct Monte Carlo integral estimate

$$\widehat{E} \{ h(Y) \} = S^{-1} \sum_{s=1}^S h(y^s)$$

where  $y^1, \dots, y^S$  are  $S$  independent pseudorandom numbers from the density  $g(y)$ , obtained by using methods described earlier. This method works if  $E \{ h(Y) \}$  exists and  $S \rightarrow \infty$ .

This method can be applied to both definite and indefinite integrals. It has the added advantage of being immediately applicable to multidimensional integrals, provided we can draw from the appropriate multivariate distribution. It has the disadvantage that it will always provide an estimate, even if the integral does not exist. For example, to obtain  $E(Y)$  for the Cauchy distribution, we could average  $S$  draws from the Cauchy. But this would be wrong because the mean of the Cauchy does not exist.

As an example, we consider the computation of  $E[\exp\{-\exp(Y)\}]$  when  $y \sim N(0, 1)$ . This is the integral:

$$E[\exp\{-\exp(Y)\}] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\{-\exp(y)\} \exp(-y^2/2) dy$$

It has no closed-form solution but can be proved to exist. We use the estimate

$$\hat{E}[\exp\{-\exp(Y)\}] = \frac{1}{S} \sum_{s=1}^S \exp\{-\exp(y^s)\}$$

where  $y^s$  is the  $s$ th draw of  $S$  draws from the  $N(0, 1)$  distribution.

This approximation task can be accomplished for a specified value of  $S$ , say, 100, by using the following code.

```
. * Integral evaluation by Monte Carlo simulation with S=100
. clear all
. qui set obs 100
. set seed 10101
. generate double y = runiform()
. generate double gy = exp(-exp(y))
. qui summarize gy, meanonly
. scalar Egy = r(mean)
. display "After 100 draws the MC estimate of E[exp{-exp(x)}] is " Egy
After 100 draws the MC estimate of E[exp{-exp(x)}] is .42950987
```

The Monte Carlo estimate of the integral is 0.430, based on 100 draws.

### 5.5.3 Monte Carlo integration using different S

It is not known in advance what value of  $S$  will yield a good Monte Carlo approximation to the integral. We can compare the outcome for several

different values of  $S$  (including  $S = 100$ ), stopping when the estimates stabilize.

To investigate this, we replace the preceding code by a Stata program that has as an argument  $S$ , the number of simulations. The program can then be called and run several times with different values of  $S$ .

The program is named `mcintegration`. The number of simulations is passed to the program as a named positional argument, `numsim`s. This variable is a local variable within the program that needs to be referenced using quotes. The call to the program needs to include a value for `numsim`s. Appendix [A.2](#) provides the details on writing a Stata program. The program is r-class and returns results for a single scalar,  $E\{g(y)\}$ , where  $g(y) = \exp\{-\exp(y)\}$ .

```
. * Program mcintegration to compute E{g(y)} numsim times
. program mcintegration, rclass
1.    version 17
2.    args numsim          // Call to program will include value for numsim
3.    drop _all
4.    qui set obs `numsim'
5.    set seed 10101
6.    generate double y = rnormal(0)
7.    generate double gy = exp(-exp(y))
8.    qui summarize gy
9.    scalar Egy = r(mean)
10.   scalar seEgy = r(sd)/sqrt(`numsim')
11.   display "#sims:" %7.0f `numsim' " MC estimate is " Egy
>           " Standard error is " seEgy
12. end
```

The program is then run several times, for  $S = 10, 100, 1000, 10000$ , and  $100000$ .

```

. * Run program mcintegration S = 10, 100, ..., 100000 times
. mcintegration 10
#sims:    10  MC estimate is .26308196  Standard error is .08293082
. mcintegration 100
#sims:   100  MC estimate is .36566172  Standard error is .02711514
. mcintegration 1000
#sims:  1000  MC estimate is .37957876  Standard error is .00837261
. mcintegration 10000
#sims: 10000  MC estimate is .38065078  Standard error is .00267433
. mcintegration 100000
#sims: 100000  MC estimate is .38162497  Standard error is .00084048

```

The estimates of  $E\{g(y)\}$  stabilize as  $S \rightarrow \infty$ , but even with  $S = 10^5$ , the estimate changes in the third decimal place. The standard error of the estimate is declining with the number of simulations, but even with 100,000 draws, the standard error of 0.00084048 equals 0.22% of the estimated value of 0.38162497.

#### 5.5.4 Halton and Hammersley sequences

It can be better to base uniform draws on deterministic sequences such as Halton and Hammersley sequences. These lead to draws with fairly even coverage over the domain of the sampling distribution. Then simulated probabilities vary less over observations relative to those calculated with random draws. This is similar to deterministic evaluation of an integral over a specified grid.

Furthermore, the sequential draws are negatively correlated. This has the advantage of reducing the variance of simulation-based estimates because  $\text{Var}(X + Y)$  equals  $\text{Var}(X) + \text{Var}(Y)$  if  $X$  and  $Y$  are independent but is less than  $\text{Var}(X) + \text{Var}(Y)$  if  $X$  and  $Y$  are negatively correlated.

The Halton sequence based on the prime number 2 is constructed as follows. Divide the unit interval  $(0, 1)$  into two parts. The dividing point  $1/2$  becomes the first element of the Halton sequence. Next, divide each part into two more parts. The dividing points,  $1/4$  and  $3/4$ , become the next two elements of the sequence. Divide each of the four parts into two parts each, and continue to obtain the sequence  $\{1/2, 1/4, 3/4, 1/8, 3/8, \dots\}$ . For a  $p$ -

dimensional Halton sequence, the starting bases are the first  $p$  primes greater than 1, specifically 2, 3, 5, 7, 11, ....

Halton sequences can be obtained using the Mata function `halton()`. The following example obtains sequences of length 200 for two variables and lists the first 8 values.

```
. * Generate 2-dimensional Halton sequences of length 200
. clear
. qui set obs 200
. mata
----- mata (type end to exit) -----
:     h = halton(200,2)
:     st_addvar("float", ("halton1", "halton2"))
      1   2
1  1   2
:     st_store(., ("halton1", "halton2"),h[.,.])
: end
-----
. list halton1 halton2 in 1/8, clean
    halton1    halton2
1.        .5    .3333333
2.        .25   .6666667
3.        .75   .1111111
4.        .125  .4444444
5.        .625  .7777778
6.        .375  .2222222
7.        .875  .5555556
8.        .0625 .8888889
```

The two sequences have means and standard deviations close to 0.5 and  $\sqrt{1/12} = 0.289$  for the uniform distribution. But the sequential draws are highly negatively correlated, as is illustrated for `halton1`.

```

. * Summary statistics and serial correlation for Halton sequences
. summarize halton1 halton2

```

Variable	Obs	Mean	Std. dev.	Min	Max
halton1	200	.4945898	.2888488	.0039063	.9921875
halton2	200	.4945679	.2879916	.0041152	.9917696

```

. qui generate lagholt1 = halton1[_n-1]
. correlate halton1 lagholt1
(obs=199)

```

	halton1 lagholt1
halton1	1.0000
lagholt1	-0.7185 1.0000

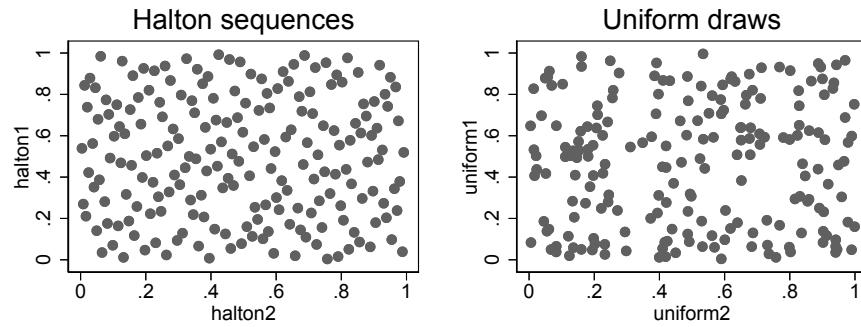
The attraction of the Halton sequences is that they provide more even coverage of the sampling distribution than do the pseudorandom uniform draws. We have

```

. * Compare deterministic Halton draws to pseudo-random uniform draws
. set seed 10101
. gen uniform1 = runiform()
. gen uniform2 = runiform()
. qui graph twoway (scatter halton1 halton2), title("Halton sequences")
>     saving(graph1.gph, replace)
. qui graph twoway (scatter uniform1 uniform2), title("Uniform draws")
>     saving(graph2.gph, replace)
. qui graph combine graph1.gph graph2.gph, ysize(2.5) xsize(6) iscale(1.5)

```

Figure 5.4 shows that there is more even coverage for the Halton sequences (panel 1) than for the draws obtained using the usual uniform generator (panel 2). The coverage for the Halton sequences becomes even more uniform as the length of the sequence increases.



**Figure 5.4.** Halton sequence compared with uniform draws

Halton sequences are best used for low-dimension sequences, say, less than 10 variables. A variation is the Hammersley sequence, which for sequences of length  $n$  sets the first variable to  $\{1/2n, 3/2n, \dots, (2n - 1)/2n\}$  and sets the remaining variables to the Halton sequence values already described.

## 5.6 Simulation for regression: Introduction

The simplest use of simulation methods is to generate a single dataset and estimate the DGP parameter  $\theta$ . Under some assumptions, if the estimated parameter  $\hat{\theta}$  differs from  $\theta$  for a large sample size, the estimator is probably inconsistent. We defer an example of this simpler simulation to section [5.6.4](#).

More often,  $\theta$  is estimated from each of  $S$ -generated datasets, and the estimates are stored and summarized to learn about the distribution of  $\hat{\theta}$  for a given DGP. For example, this approach is necessary if one wants to check the validity of a standard error estimator or the finite-sample size of a test. This approach requires the ability to perform the same analysis  $S$  times and to store the results from each simulation. The simplest approach is to write a Stata program for the analysis of one simulation and then use `simulate` to run this program many times.

### 5.6.1 Simulation example: OLS with $\chi^2$ errors

In this section, we use simulation methods to investigate the finite-sample properties of the OLS estimator with random regressors and skewed errors. If the errors are i.i.d., the fact that they are skewed has no effect on the large-sample properties of the OLS estimator. However, when the errors are skewed, we will need a larger sample size for the asymptotic distribution to better approximate the finite-sample distribution of the OLS estimator than when the errors are normal. This example also highlights an important modeling decision: when  $y$  is skewed, we sometimes choose to model  $E(\ln y|\mathbf{x})$  instead of  $E(y|\mathbf{x})$  because we believe the disturbances enter multiplicatively instead of additively. This choice is driven by the multiplicative way the error affects the outcome and is independent of the functional form of its distribution. As illustrated in this simulation, the asymptotic theory for the OLS estimator works well when the errors are i.i.d. from a skewed distribution.

We consider the following DGP,

$$y = \beta_1 + \beta_2 x + u; \quad u \sim \chi^2(1) - 1; \quad x \sim \chi^2(1)$$

where  $\beta_1 = 1$ ,  $\beta_2 = 2$ , and the sample size  $N = 150$ . For this DGP, the error  $u$  is independent of the regressor  $x$  (ensuring consistency of OLS) and has a mean of 0, variance of 2, skewness of  $\sqrt{8}$ , and kurtosis of 15. By contrast, a normal error has a skewness of 0 and a kurtosis of 3.

We wish to perform 1,000 simulations, where in each simulation we obtain parameter estimates, standard errors,  $t$ -values for the  $t$  test of  $H_0: \beta_2 = 2$ , and the outcome of a two-sided test of  $H_0$  at level 0.05. Inference throughout this example is based on default OLS standard errors.

Two of the most frequently changed parameters in a simulation study are the sample size and the number of simulations. Thus, these two parameters are almost always stored in something that can easily be changed. We use global macros. In the output below, we store the number of observations in the global macro `numobs` and the number of repetitions in the global macro `numsims`. We use these global macros in the examples in this section.

```
. * Define global macros for sample size and number of simulations
. global numobs 150           // Sample size N
. global numsims "1000"       // Number of simulations
```

We first write the `chi2data` program, which generates data on  $y$ , performs OLS, and returns  $\hat{\beta}_2$ ,  $s_{\hat{\beta}_2}$ ,  $t_2 = (\hat{\beta}_2 - 2)/s_{\hat{\beta}_2}$ , a rejection indicator  $r_2 = 1$  if  $|t_2| > t_{0.025}(148)$ , and the  $p$ -value for the two-sided  $t$  test. The `chi2data` program is an r-class program, so these results are returned in `r()` using the `return` command.

```

. * Program for finite-sample properties of OLS
. program chi2data, rclass
1.     version 17
2.     drop _all
3.     set obs $numobs
4.     generate double x = rchi2(1)
5.     generate y = 1 + 2*x + rchi2(1)-1      // Demeaned chi^2 error
6.     regress y x
7.     return scalar b2 = _b[x]
8.     return scalar se2 = _se[x]
9.     return scalar t2 = (_b[x]-2)/_se[x]
10.    return scalar r2 = abs(return(t2))>invtail($numobs-2,.025)
11.    return scalar p2 = 2*ttail($numobs-2,abs(return(t2)))
12. end

```

This code is used to produce all the statistics analyzed below, including the  $t$  statistic plotted in figure 5.5. Monte Carlo studies of tests typically consider test rejection rates and  $p$ -values, rather than the test statistic itself. In that case, we can instead use the `test` command, which computes an  $F$  statistic with the same  $p$ -value, and reject at 5% if  $p < 0.05$ . Subsequent Monte Carlos in this section and, for example, section 11.7 use the `test` command.

The following output illustrates that `test` and the manual calculations yield the same  $p$ -value.

```

. * An F test gives same p-value as the manual t test in program chi2data
. set seed 10101
. qui chi2data
. return list
scalars:
r(p2) = .3076722413709826
r(r2) = 0
r(t2) = 1.023647108363717
r(se2) = .0741344926219221
r(b2) = 2.075887559002442

. qui test x=2
. return list
scalars:
r(drop) = 0
r(df_r) = 148
r(F) = 1.0478534024614
r(df) = 1
r(p) = .3076722413709826

```

We use `simulate` to call `chi2data` `$numsims` times and to store the results; here `$numsims = 1000`. The current dataset is replaced by one with the results from each simulation. These results can be displayed by using `summarize`, where `obs` in the output refers to the number of simulations and not the sample size in each simulation.

```
. * Simulation for finite-sample properties of OLS
. simulate b2f=r(b2) se2f=r(se2) t2f=r(t2) reject2f=r(r2) p2f=r(p2),
>     seed(10101) reps($numsims) nolegend nodots: chi2data

. summarize b2f se2f reject2f
```

Variable	Obs	Mean	Std. dev.	Min	Max
b2f	1,000	2.003871	.0878981	1.788395	2.504547
se2f	1,000	.0841427	.0180844	.0388639	.1792164
reject2f	1,000	.05	.218054	0	1

The `summarize` output indicates that 1) the mean of the point estimates is very close to the true value of 2, 2) the standard deviation of the point estimates is close to the mean of the standard errors, and 3) the rejection rate of 0.05 coincides with the size of 0.05. The exact result for the rejection rate is coincidental—with a different seed, we do not get exactly 0.05.

Further information on the distribution of the results can be obtained by using the `mean`, the `summarize`, `detail`, and the `kdensity` commands.

### 5.6.2 Interpreting simulation output

We consider in turn unbiasedness of  $\hat{\beta}_2$ , correctness of the standard error formula for  $s_{\hat{\beta}_2}$ , distribution of the  $t$  statistic, and test size.

Because interest lies in the averages over the simulations of the various statistics, we use the `mean` command.

```
. * Report results for simulation averages
. mean b2f se2f reject2f
```

Mean estimation Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
b2f	2.003871	.0027796	1.998416 2.009325
se2f	.0841427	.0005719	.0830204 .0852649
reject2f	.05	.0068955	.0364687 .0635313

### Unbiasedness of estimator

The average of  $\hat{\beta}_2$  over the 1,000 estimates,  $\bar{\hat{\beta}}_2 = (1/1000) \sum_{s=1}^{1000} \hat{\beta}_s$ , is the simulation estimate of  $E(\hat{\beta}_2)$ . Here  $\bar{\hat{\beta}}_2 = 2.004$  (see the mean of `b2f`) is very close to the DGP value  $\beta_2 = 2.0$ , suggesting that the estimator is unbiased.

However, this comparison should account for simulation error. From the `mean` command, the simulation yields a 95% confidence interval for  $E(\hat{\beta}_2)$  of [1.998, 2.009]. This interval is quite narrow and includes 2.0, so we conclude that  $E(\hat{\beta}_2)$  is unbiased. Note, however, that if  $\hat{\beta}_2$  is unbiased, then in 5% of such simulation exercises, this 95% confidence interval will not include the DGP value of  $\beta_2$ .

Many estimators, particularly nonlinear estimators, are biased in finite samples. Then exercises such as this can be used to estimate the magnitude of the bias in typical sample sizes. If the estimator is consistent, then any bias should disappear as the sample size  $N$  goes to infinity.

### Standard errors

The variance of  $\hat{\beta}_2$  over the 1,000 estimates,  $s_{\hat{\beta}_2}^2 = (1/999) \sum_{s=1}^{1000} (\hat{\beta}_s - \bar{\hat{\beta}}_2)^2$ , is the simulation estimate of  $\sigma_{\hat{\beta}_2}^2 = \text{Var}(\hat{\beta}_2)$ , the variance of  $\hat{\beta}_2$ .

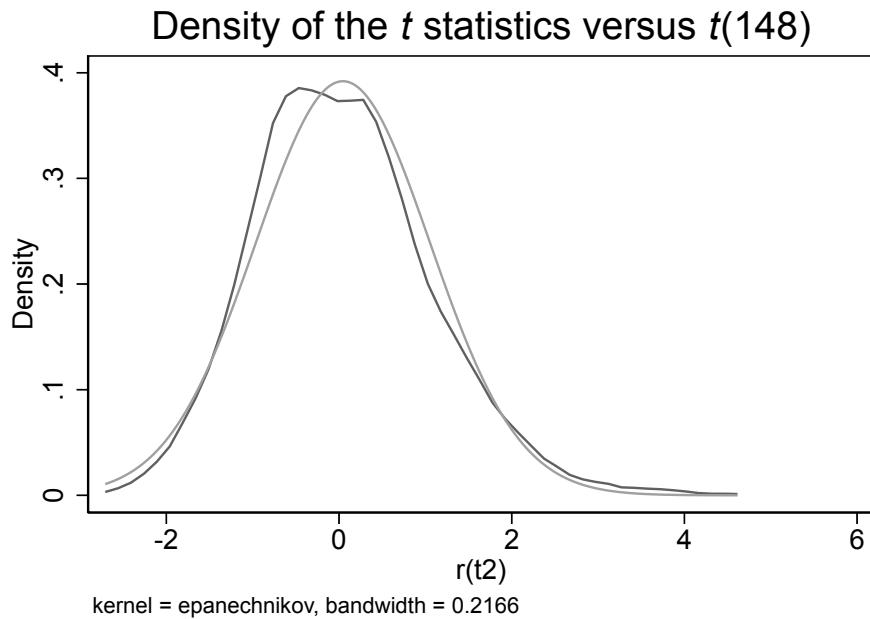
Here the average of the standard errors across the simulations is  $\text{se}(\hat{\beta}_2) = 0.084$  (see the mean of `se2f`), with 95% confidence interval for  $\text{se}(\hat{\beta}_2)$  of [0.083, 0.085]. Because this interval does not include  $s_{\hat{\beta}_2} = 0.088$

(see the standard deviation of  $b2f$ ), there is evidence that  $\text{se}(\hat{\beta}_2)$  is slightly biased for  $\sigma_{\hat{\beta}_2}$ , so the asymptotic distribution is not perfectly approximating the finite-sample distribution. Note that for the current DGP,  $\{\text{se}(\hat{\beta}_2)\}^2$  is unbiased for  $\sigma_{\hat{\beta}_2}^2$ , but this does not imply that upon taking the square root,  $\text{se}(\hat{\beta}_2)$  is unbiased for  $\sigma_{\hat{\beta}_2}$ .

### t statistic

Because we impose looser restrictions on the DGP,  $t$  statistics are not exactly  $t$  distributed, and  $z$  statistics are not exactly  $z$  distributed. However, the extent to which they diverge from the reference distribution disappears as the sample size increases. The output below generates the graph in figure 5.5, which compares the density of the  $t$  statistics with the  $t(148)$  distribution.

```
. * Plot the density of the t statistics and compare with theoretical t(148)
. kdensity t2f, student(148) legend(off) scale(1.2)
> title("Density of the {it:t} statistics versus {it:t}(148)")
```



**Figure 5.5.**  $t$  statistic density compared with theoretical  $t(148)$

Though the graph highlights some differences between the finite-sample and the asymptotic distributions, the divergence between the two does not appear to be great. From output not given, `centile t2f, centile(2.5 97.5)` yields values – 1.732 and 2.041 compared with – 1.976 and 1.976 for the  $t(148)$  distribution.

Rather than focus on the distribution of the  $t$  statistics, we instead focus on the size of tests or coverage of confidence intervals based on these statistics.

#### Test size

The size of the test is the probability of rejecting  $H_0$  when  $H_0$  is true. Because the DGP sets  $\beta_2 = 2$ , we consider a two-sided test of  $H_0: \beta_2 = 2$  against  $H_a: \beta_2 \neq 2$ . The level or nominal size of the test is set to 0.05, and the  $t$  test is used. The proportion of simulations that leads to a rejection of  $H_0$  is known as the rejection rate, and this proportion is the simulation estimate of the true test size.

Here the estimated rejection rate is 0.050 (see the mean of `reject2f`). The associated 95% confidence interval (from `mean reject2f`) is [0.036, 0.064], which is quite wide but includes 0.05. The width of this confidence interval is partially a result of having run only 1,000 repetitions and partially an indication that, with 150 observations, the true size of the test can differ from the nominal size. When this simulation is rerun with 10,000 repetitions, the estimated rejection rate is 0.048, and the confidence interval is [0.044, 0.052].

The simulation results also include the variable `p2f`, which stores the  $p$ -values of each test. If the  $t(148)$  distribution is the correct distribution for the  $t$  test, then `p2f` should be uniformly distributed on (0, 1). A histogram, not shown, reveals this to be the case.

More simulations are needed to accurately measure test size (and power) than are needed for bias and standard error calculations. For a test with estimated size  $a$  based on  $S$  simulations, a 95% confidence interval for the true size is  $a \pm 1.96 \times \sqrt{a(1 - a)/S}$ . For example, if  $a = 0.06$  and  $S = 10000$ , then the 95% confidence interval is [0.055, 0.065]. A more

detailed Monte Carlo experiment for test size and power is given in section [11.7](#).

#### **Number of simulations**

Ideally, 10,000 simulations or more would be run in reported results, but this can be computationally expensive. With only 1,000 simulations, there can be considerable simulation noise, especially for estimates of test size (and power).

#### **5.6.3 Variations**

The preceding code is easily adapted to other problems of interest.

##### **Different sample size and number of simulations**

Sample size can be changed by changing the global macro `numobs`. Many simulation studies focus on finite-sample deviations from asymptotic theory. For some estimators, most notably IV with weak instruments, such deviations can occur even with samples of many thousands of observations.

Changing the global macro `numsims` can increase the number of simulations to yield more precise simulation results.

##### **Test power**

A type II error occurs if a test fails to reject  $H_0$  when  $H_0$  is false. The power is one minus the probability of making this error, so  
 $\text{power} = \Pr(\text{reject } H_0 | H_0 \text{ false})$ . The larger the difference between the tested value and the true value, the greater the power and the rejection rate. The example below modifies `chi2data` to estimate the power of a test against the false null hypothesis that  $\beta_2 = 2$ ; instead,  $\beta_2 = 2.1$ .

```

. * Program for finite-sample properties of OLS: power
. program chi2datab, rclass
1.     version 17
2.     drop _all
3.     set obs $numobs
4.     generate double x = rchi2(1)
5.     generate y = 1 + 2.1*x + rchi2(1)-1      // Demeaned chi^2 error
6.     regress y x
7.     return scalar b2 = _b[x]
8.     return scalar se2 = _se[x]
9.     test x=2
10.    return scalar r2 = (r(p)<.05)
11. end

```

Below, we use `simulate` to run the simulation 1,000 times, and then we summarize the results.

```

. * Power simulation for finite-sample properties of OLS
. simulate b2f=r(b2) se2f=r(se2) reject2f=r(r2),
>     seed(10101) reps($num sims) nolegend nodots: chi2datab

. mean b2f se2f reject2f
Mean estimation                                         Number of obs = 1,000

```

	Mean	Std. err.	[95% conf. interval]
b2f	2.103871	.0027796	2.098416 2.109325
se2f	.0841427	.0005719	.0830204 .0852649
reject2f	.237	.0134541	.2105985 .2634015

The sample mean of `reject2f` provides an estimate of the power. The estimated power is 0.237, which is not high. Increasing the sample size or the distance between the tested value and the true value will increase the power of the test.

A useful way to incorporate power estimation is to define the hypothesized value of  $\beta_2$  to be an argument of the program `chi2datab`. This is demonstrated in the more detailed Monte Carlo experiment in section [11.7](#).

### Different error distributions

We can investigate the effect of using other error distributions by changing the distribution used in `chi2data`. For linear regression, the  $t$  statistic

becomes closer to  $t$  distributed as the error distribution becomes closer to i.i.d. normal. For nonlinear models, the exact finite-sample distribution of estimators and test statistics is unknown even if the errors are i.i.d. normal.

The example in section [5.6.2](#) used draws of both regressors and errors that differed in each simulation. This corresponds to simple random sampling where we jointly sample the pair  $(y, x)$ , especially relevant to survey data where individuals are sampled, and we use data  $(y, x)$  for the sampled individuals. An alternative approach is that of fixed regressors in repeated trials, especially relevant to designed experiments. Then we draw a sample of  $x$  only once, and we use the same sample of  $x$  in each simulation while redrawing only the error  $u$  (and hence  $y$ ). In that case, we create `fixedx.dta`, which has 150 observations on a variable,  $x$ , that is drawn from the  $\chi^2(1)$  distribution, and we replace lines 2–4 of `chi2data` by typing `use fixedx, clear`.

#### 5.6.4 Estimator consistency or inconsistency

Establishing estimator consistency or inconsistency requires less coding because we need to generate data and obtain estimates only once, with a large  $N$ , and then compare the estimates with the DGP values.

For the preceding DGP and estimator with  $N = 10000$ , we obtain

```
. * Consistency of OLS in preceding simulation setup
. clear
. qui set obs 10000
. set seed 10101
. generate double x = rchi2(1)
. generate y = 1 + 2*x + rchi2(1)-1      // Demeaned chi^2 error
. regress y x, noheader
```

	y	Coefficient	Std. err.	t	P> t	[95% conf. interval]
	x	2.003907	.0099924	200.54	0.000	1.98432 2.023494
	_cons	.9716907	.0169879	57.20	0.000	.938391 1.00499

The intercept and slope coefficients are very close to their DGP values of 1.0 and 2.0. As expected, the OLS estimator appears to be consistent.

Next we consider a classical errors-in-variables model of measurement error. Here not only the inconsistency of the OLS estimator is known but also the magnitude of the inconsistency, so we have a benchmark for comparison.

The DGP considered is

$$y = \beta x^* + u; \quad x^* \sim N(0, 9); \quad u \sim N(0, 1)$$

$$x = x^* + v; \quad v \sim N(0, 1)$$

OLS regression of  $y$  on  $x^*$  consistently estimates  $\beta$ . However, only data on  $x$  rather than  $x^*$  are available, so we instead obtain  $\hat{\beta}$  from an OLS regression of  $y$  on  $x$ .

It is a well-known result that then  $\hat{\beta}$  is inconsistent, with a downward bias,  $s\beta$ , where  $s = \sigma_v^2 / (\sigma_v^2 + \sigma_{x^*}^2)$  is the noise–signal ratio; see [Cameron and Trivedi \(2005, 903\)](#). For the DGP under consideration, this ratio is  $1/(1+9) = 0.1$ , so  $\text{plim } \hat{\beta} = \beta - s\beta = 1 - 0.1 \times 1 = 0.9$ .

The following simulation checks this theoretical prediction, with sample size set to 10,000. We use `drawnorm` to jointly draw  $(x^*, u, v)$ , though we could have more simply made three separate standard normal draws. We set  $\beta = 1$ .

```
. * Inconsistency of OLS in errors-in-variables model (measurement error)
. clear
. qui set obs 10000
. set seed 10101
. matrix mu = (0,0,0)
. matrix sigmasq = (9,0,0\0,1,0\0,0,1)
. drawnorm xstar u v, means(mu) cov(sigmasq)
. generate y = 1*xstar + u // DGP for y depends on xstar
. generate x = xstar + v // x is mismeasured xstar
. regress y x, noconstant noheader
```

		Coefficient	Std. err.	t	P> t	[95% conf. interval]
	y					
	x	.899366	.0043202	208.17	0.000	.8908974 .9078346

The OLS estimate is very precisely estimated, given the large sample size. The estimate of 0.8994 clearly differs from the DGP value of 1.0, so OLS is inconsistent. Furthermore, the simulation estimate essentially equals the theoretical value of 0.9.

### 5.6.5 Simulation with endogenous regressors

Endogeneity is one of the most frequent causes of estimator inconsistency. A simple method to generate an endogenous regressor is to first generate the error  $u$  and then generate the regressor  $x$  to be the sum of a multiple of  $u$  and an independent component.

We adapt the previous DGP as follows:

$$y = \beta_1 + \beta_2 x + u; \quad u \sim N(0, 1);$$

$$x = z + u + v; \quad z \sim N(0, 1); \quad v \sim N(0, 1)$$

We set  $\beta_1 = 10$  and  $\beta_2 = 2$ . For this DGP, the correlation between  $x$  and  $u$  equals 0.5. We let  $N = 150$ .

The following program generates the data and estimates:

```
. * Program for OLS with endogenous regressor
. clear
. program endogreg, rclass
1.      version 17
2.      drop _all
3.      set obs $numobs
4.      generate u = rnormal(0)
5.      generate z = rnormal()
6.      generate v = rnormal(0)
7.      generate x = z + u + v // Endogenous regressor
8.      generate y = 10 + 2*x + u
9.      regress y x
10.     return scalar b2 = _b[x]
11.     return scalar se2 = _se[x]
12.     test x=0
13.     return scalar r2 = (r(p)<.05)
14. end
```

We run the simulations and summarize the results.

```

. * Simulation for OLS with endogenous regressor
. simulate b2r=r(b2) se2r=r(se2) reject2r=r(r2),
>      seed(10101) reps($numsim) nolegend nodots: endogreg

. mean b2r se2r reject2r

```

Mean estimation Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
b2r	2.334661	.0012874	2.332134 2.337187
se2r	.0386734	.0000976	.038482 .0388649
reject2r	1	0	.

These 1,000 repetitions indicate that for  $N = 150$ , the OLS estimator is biased by about 17%, the reported standard error is about 5% too small,  $0.03867/(0.001287 \times \sqrt{1000}) = 0.9502$ , and we always reject the true null hypothesis that  $\beta_2 = 2$ .

By setting  $N$  large, we could also show that the OLS estimator is inconsistent with a single repetition. As a variation, we could instead estimate by IV, with  $z$  a valid instrument for  $x$  given this DGP, and verify that the IV estimator is consistent.

## 5.7 Additional resources

The key reference for random-number functions is `help random number functions`. This covers most of the generators illustrated in this chapter and several other standard ones that have not been used. Note, however, that the `rnbinoomial(k, p)` function for making draws from the negative binomial distribution has a different parameterization from that used in this book. The key Stata commands for simulation are `simulate` and `postfile` (see [R] **simulate** and [P] **postfile**). The `simulate` prefix requires first collecting commands into a program; see [P] **program**.

A standard book that presents algorithms for random-number generation is Press et al. (2007). Cameron and Trivedi (2005) discuss random-number generation and present a Monte Carlo study. Simulations are used throughout the current book; see especially the chapters on Bayesian methods, testing methods, and bootstrap methods.

## 5.8 Exercises

1. Give the commands `set obs 100` and `generate u=runiform()` and `summarize`. Then, give command `clear` and repeat the initial commands. Did you get the same result? If not, explain why, and adapt your code to get the same results.
2. Using the normal generator, generate a random draw from a 50–50 scale mixture of  $N(1, 1)$  and  $N(1, 3^2)$  distributions. Repeat the exercise with the  $N(1, 3^2)$  component replaced by  $N(3, 1)$ . For both cases, display the features of the generated data by using a kernel density plot.
3. Generate 1,000 observations from the  $F(5, 10)$  distribution. Use `rchi2()` to obtain draws from the  $\chi^2(5)$  and the  $\chi^2(10)$  distributions. Compare the sample moments with their theoretical counterparts.
4. Make 1,000 draws from the  $N(6, 2^2)$  distribution by making a transformation of draws from  $N(0, 1)$  and then making the transformation  $Y = \mu + \sigma Z$ .
5. Generate 1,000 draws from the  $t(6)$  distribution, which has a mean of 0 and a variance of 4. Compare your results with those from exercise 4.
6. Generate a large sample from the  $N(\mu = 1, \sigma^2 = 1)$  distribution and estimate  $\sigma/\mu$ , the coefficient of variation (cv). Verify that the sample estimate is a consistent estimate.
7. Generate a draw from a multivariate normal distribution,  $N(\boldsymbol{\mu}, \boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}'')$ , with  $\boldsymbol{\mu}' = [0\,x\,0\,x\,0]$  and

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \sqrt{3} & 0 \\ 0 & \sqrt{3} & \sqrt{6} \end{bmatrix}, \text{ or } \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 0 & 3 & 9 \end{bmatrix}$$

using transformations based on this Cholesky decomposition. Compare your results with those based on using the `drawnorm` command.

8. Let  $s$  denote the sample estimate of  $\sigma$  and  $\bar{x}$  denote the sample estimate of  $\mu$ . The cv  $\sigma/\mu$ , which is the ratio of the standard deviation

to the mean, is a dimensionless measure of dispersion. The asymptotic distribution of the sample  $\text{cv } s/\bar{x}$  is

$N[\sigma/\mu, (N - 2)^{-1/2}(\sigma/\mu)^2\{0.5 + (\sigma/\mu)^2\}]$ ; see [Miller \(1991\)](#). For  $N = 25$ , using either `simulate` or `postfile`, compare the Monte Carlo and asymptotic variance of the sample  $\text{cv}$  with the following specification of the DGP:  $x \sim N(\mu, \sigma^2)$  with three different values of  $\text{CV} = 0.1, 0.33$ , and  $0.67$ .

9. It is suspected that making draws from the truncated normal using the method given in section [5.4.4](#) may not work well when sampling from the extreme tails of the normal. Using different truncation points, check this suggestion.
10. Repeat the example of section [5.6.1](#) (OLS with  $\chi^2$  errors), now using the `postfile` command. Use `postfile` to save the estimated slope coefficient, standard error, the  $t$  statistic for  $H_0: \beta = 2$ , and an indicator for whether  $H_0$  is rejected at the 0.05 level in a Stata file named `simresults`. The template program is as follows:

```

* Postfile and post example: repeat OLS with chi-squared errors example
clear
set seed 10101
program simbypost
    version 17
    tempfile simfile
    postfile `simfile' b2 se2 t2 reject2 p2 using simresults, replace
    quietly {
        forvalues i = 1/1000 {
            drop _all
            set obs 150
            generate x = rchi2(1)
            generate y = 1 + 2*x + rchi2(1) - 1 // demeaned chi^2 error
            regress y x
            scalar b2 = _b[x]
            scalar se2 = _se[x]
            scalar t2 = (_b[x]-2)/_se[x]
            scalar reject2 = abs(t2) > invttail(1000-2,.025)
            scalar p2 = 2*ttail(1000-2,abs(t2))
            post `simfile' (b2) (se2) (t2) (reject2) (p2)
        }
    }
    postclose `simfile'
end
simbypost
use simresults, clear
summarize

```



# **Chapter 6**

## **Linear regression with correlated errors**

## 6.1 Introduction

Chapter 3 presented ordinary least-squares (OLS) estimation with inference based on either heteroskedasticity-robust or cluster-robust standard errors. Now we go further and introduce feasible generalized least-squares (FGLS) estimation based on a richer specification of the model for the error. This can lead to more efficient estimation than OLS estimation, leading to smaller standard errors, narrower confidence intervals, and larger  $t$  statistics.

Much of the chapter focuses on the case of model errors that are clustered, with correlation for observations in the same cluster and independence for observations in different clusters. This is a common complication in microeconomic studies with cross-sectional data, the focus of this chapter, and with panel data, presented in chapter 8. Failure to appropriately control for clustered errors can lead to very erroneous statistical inference.

There are several ways to proceed when model errors are clustered. A common approach is to simply use the OLS estimator and forgo any of the potential efficiency gains of alternative estimators. The random-effects (RE) estimator introduces a purely random cluster-specific effect and estimates by FGLS. The fixed-effects (FE) estimator also introduces cluster-specific effects but relaxes the assumption that these are purely random. Mixed models or hierarchical models permit richer within-cluster correlation structures than the RE model. For all of these approaches, one can obtain cluster-robust standard errors that guard against misspecification of the model for error correlation. The biggest distinction is that the FE estimator enables consistent parameter estimation under assumptions weaker than those needed for consistency of OLS and RE estimators.

The chapter begins with a review of FGLS and application to cross-sectional data with heteroskedastic errors. Four sections of the chapter then detail the different models and inference methods available when model errors are clustered. Even though we consider a cross-sectional data example, the RE and FE estimators are most easily estimated using the `xt` commands developed for panel data. The chapter then considers

multiequation seemingly unrelated regressions (SUR), a different example of correlated errors.

Other linear model examples of FGLS include the population-averaged estimator and the RE estimator for panel data (sections [8.4](#) and [8.7](#)) and the three-stage least-squares estimator for simultaneous-equations systems (section [7.10](#)). Many of the methods carry over to nonlinear regression models, where efficient estimation is by ML rather than FGLS and stronger distributional assumptions may be required for estimator consistency. Extension to systems of nonlinear equations is given in section 18.11.2.

This chapter concludes with a stand-alone presentation of a quite distinct topic: survey estimation methods that explicitly control for the three complications of data from complex surveys—sampling that is weighted, clustered, and stratified.

## 6.2 Generalized least-squares and FGLS regression

We provide an overview of theory for generalized least-squares (GLS) and FGLS estimation.

### 6.2.1 GLS for heteroskedastic errors

A simple example is the single-equation linear regression model with heteroskedastic independent errors, where a specific model for heteroskedasticity is given. Specifically,

$$\begin{aligned} y_i &= \mathbf{x}'_i \boldsymbol{\beta} + u_i, \quad i = 1, \dots, N \\ u_i &= \sigma(\mathbf{z}_i) \varepsilon_i \end{aligned} \tag{6.1}$$

where  $\varepsilon_i$  satisfies  $E(\varepsilon_i | \mathbf{x}_i, \mathbf{z}_i) = 0$ ,  $E(\varepsilon_i \varepsilon_j | \mathbf{x}_i, \mathbf{z}_i, \mathbf{x}_j, \mathbf{z}_j) = 0$ ,  $i \neq j$ , and  $E(\varepsilon_i^2 | \mathbf{x}_i, \mathbf{z}_i) = 1$ . The function  $\sigma(\mathbf{z}_i)$ , called a skedasticity function, is a specified scalar-valued function of the observable variables  $\mathbf{z}_i$ . The special case of homoskedastic regression arises if  $\sigma(\mathbf{z}_i) = \sigma$ , a constant. The elements of the vectors  $\mathbf{z}$  and  $\mathbf{x}$  may or may not overlap.

Under these assumptions, the errors  $u_i$  in (6.1) have zero mean and are uncorrelated but are heteroskedastic with variance  $\sigma^2(\mathbf{z}_i)$ . Then OLS estimation of (6.1) yields consistent estimates, but more efficient estimation is possible if we instead estimate by OLS a transformed model that has homoskedastic errors. Transforming the model by multiplying by  $w_i = 1/\sigma(\mathbf{z}_i)$  yields the homoskedastic regression

$$\left( \frac{y_i}{\sigma(\mathbf{z}_i)} \right) = \left( \frac{\mathbf{x}_i}{\sigma(\mathbf{z}_i)} \right)' \boldsymbol{\beta} + \varepsilon_i \tag{6.2}$$

because  $u_i/\sigma(\mathbf{z}_i) = \{\sigma(\mathbf{z}_i)\varepsilon_i\}/\sigma(\mathbf{z}_i) = \varepsilon_i$  and  $\varepsilon_i$  is homoskedastic. The GLS estimator is the OLS estimator of this transformed model. This regression can also be interpreted as a weighted linear regression of  $y_i$  on  $\mathbf{x}_i$  with the weight  $w_i = 1/\sigma(\mathbf{z}_i)$  assigned to the  $i$ th observation. In practice,  $\sigma(\mathbf{z}_i)$  may

depend on unknown parameters, leading to the FGLS estimator, which uses the estimated weights  $\widehat{\sigma}(\mathbf{z}_i)$  as explained below.

### 6.2.2 GLS and FGLS

More generally, we begin with the linear model in matrix notation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

By the Gauss–Markov theorem, the OLS estimator is efficient among linear unbiased estimators if the linear regression model errors are zero-mean independent and homoskedastic.

We suppose instead that  $E(\mathbf{u}\mathbf{u}'|\mathbf{X}) = \Omega$ , where  $\Omega \neq \sigma^2\mathbf{I}$  for a variety of reasons that may include heteroskedasticity or clustering, or both. Then the efficient GLS estimator is obtained by OLS estimation of the transformed model

$$\Omega^{-1/2}\mathbf{y} = \Omega^{-1/2}\mathbf{X}\boldsymbol{\beta} + \varepsilon$$

where  $\Omega^{-1/2}\Omega\Omega^{-1/2'} = \mathbf{I}$  so that the transformed error  $\varepsilon = \Omega^{-1/2}\mathbf{u} \sim [\mathbf{0}, \mathbf{I}]$  is independent and homoskedastic. In the heteroskedastic case,  $\Omega = \text{Diag}\{\sigma^2(\mathbf{z}_i)\}$ , so  $\Omega^{-1/2} = \text{Diag}\{1/\sigma(\mathbf{z}_i)\}$ .

In practice,  $\Omega$  is not known. Instead, we specify an error variance matrix model,  $\Omega = \Omega(\gamma)$ , that depends on a finite-dimensional parameter vector  $\gamma$  and, possibly, data. Given a consistent estimate  $\widehat{\gamma}$  of  $\gamma$ , we form  $\widehat{\Omega} = \Omega(\widehat{\gamma})$ . Different situations correspond to different models for  $\Omega(\gamma)$  and estimates of  $\widehat{\Omega}$ . The FGLS estimator is the OLS estimator from the regression of  $\widehat{\Omega}^{-1/2}\mathbf{y}$  on  $\widehat{\Omega}^{-1/2}\mathbf{X}$  and equals

$$\widehat{\boldsymbol{\beta}}_{\text{FGLS}} = \left( \mathbf{X}'\widehat{\Omega}^{-1}\mathbf{X} \right)^{-1} \mathbf{X}'\widehat{\Omega}^{-1}\mathbf{y}$$

Under the assumption  $\Omega(\gamma)$  is correctly specified, the asymptotic variance–covariance matrix of the estimator (vce) of  $\hat{\beta}_{\text{FGLS}}$  is  $(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X})^{-1}$ ; this is a two-step estimator but one for which the first-step estimation (of  $\Omega$  by  $\hat{\Omega}$ ) makes no difference to the asymptotic distribution of the second-step estimator ( $\hat{\beta}_{\text{FGLS}}$ ).

### 6.2.3 Robust standard errors for FGLS

The FGLS estimator requires specification of a model,  $\Omega(\gamma)$ , for the error variance matrix. Usually, it is clear what general complication is likely to be present. For example, heteroskedastic errors are likely with cross-sectional data, but it is not clear what specific model for that complication is appropriate. If the model for  $\Omega(\gamma)$  is misspecified, then FGLS is still consistent, though it is no longer efficient. More importantly, the usual vce of  $\hat{\beta}_{\text{FGLS}}$  will be incorrect. Instead, a robust estimator of the vce should be used.

We therefore distinguish between the true error variance matrix,  $E(\mathbf{u}\mathbf{u}'|\mathbf{X})$ , and the specified model for the error variance, denoted by  $\Omega = \Omega(\gamma)$ . In the statistics literature, especially that for generalized linear models,  $\Omega(\gamma)$  is called a working variance matrix. Form the estimate  $\hat{\Omega} = \Omega(\hat{\gamma})$ , where  $\hat{\gamma}$  is an estimate of  $\gamma$ . Then, do FGLS with the weighting matrix  $\hat{\Omega}^{-1}$ , but obtain a robust estimate of the vce. This estimator is also called a weighted least-squares (wls) estimator to indicate that we no longer maintain that  $\Omega(\gamma) = E(\mathbf{u}\mathbf{u}'|\mathbf{X})$ .

Table 6.1 presents formulas for the default, heteroskedastic–robust and cluster–robust vce of the OLS and FGLS estimators. The heteroskedastic–robust vce of the FGLS estimator requires that  $\hat{\Omega}$  be diagonal and  $N \rightarrow \infty$ . The cluster–robust vce of the FGLS estimator requires that  $\hat{\Omega} = \text{Diag}(\hat{\Omega}_g)$  be block-diagonal and  $G \rightarrow \infty$ .

**Table 6.1.** OLS and FGLS estimators and their estimated variance

Estimator	Estimator definition and estimated asymptotic variance
OLS	$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ $\hat{V}_{\text{default}}(\hat{\beta}) = s^2(\mathbf{X}'\mathbf{X})^{-1}$ $\hat{V}_{\text{het}}(\hat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1}(b \times \sum_i \hat{u}_i^2 \mathbf{x}_i \mathbf{x}'_i)(\mathbf{X}'\mathbf{X})^{-1}$ $\hat{V}_{\text{cluster}}(\hat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1} \left( c \times \sum_g \mathbf{X}'_g \hat{\mathbf{u}}_g \hat{\mathbf{u}}'_g \mathbf{X}_g \right) (\mathbf{X}'\mathbf{X})^{-1}$
FGLS	$\hat{\beta} = \left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\hat{\Omega}^{-1}\mathbf{y}$ $\hat{V}_{\text{default}}(\hat{\beta}) = \left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1}$ $\hat{V}_{\text{het}}(\hat{\beta}) = \left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1}(b \times \sum_i \hat{u}_i^2 \mathbf{x}_i \mathbf{x}'_i / \hat{\omega}_{gg}) \left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1}$ $\hat{V}_{\text{cluster}}(\hat{\beta}) = \left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1} \left( c \times \sum_g \mathbf{X}'_g \hat{\Omega}_g^{-1} \hat{\mathbf{u}}_g \hat{\mathbf{u}}'_g \hat{\Omega}_g^{-1} \mathbf{X}'_g \right)$ $\left(\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X}\right)^{-1}$

notes: The cluster-robust estimate for FGLS assumes that  $\hat{\Omega} = \text{Diag}(\hat{\Omega}_g)$ , where  $g = 1, \dots, G$  defines the clusters. The degrees-of-freedom corrections are  $b = N/(N - K)$  and  $c = \{N/(N - K)\}/\{G/G - 1\}$ , where  $K$  is the number of regression parameters and  $G$  is the number of clusters.  $\hat{\omega}_{gg}$  denotes the  $g$ th diagonal entry in  $\hat{\Omega}_g$ .

Consider the case of clustered errors. The OLS estimator can be expressed as  $\hat{\beta} = \beta + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}$ , given  $\mathbf{y} = \mathbf{X}'\beta + \mathbf{u}$ . The OLS estimator therefore has asymptotic variance  $(\mathbf{X}'\mathbf{X})^{-1}E(\mathbf{X}'\mathbf{u}\mathbf{u}'\mathbf{X})(\mathbf{X}'\mathbf{X})^{-1}$ , given  $E(\mathbf{u}|\mathbf{X}) = \mathbf{0}$ . For  $g$ th cluster  $\mathbf{y}_g = \mathbf{X}'_g\beta + \mathbf{u}_g$ , we have  $E(\mathbf{X}'\mathbf{u}\mathbf{u}'\mathbf{X}) = \sum_g \sum_h E(\mathbf{X}'_g \mathbf{u}_g \mathbf{u}'_h \mathbf{X}_h)$ . Given error independence across clusters, the double sum reduces to  $\sum_g E(\mathbf{X}'_g \mathbf{u}_g \mathbf{u}'_g \mathbf{X}_g)$ , which can be estimated by  $\sum_g \mathbf{X}'_g \hat{\mathbf{u}}_g \hat{\mathbf{u}}'_g \mathbf{X}_g$ , where  $\hat{\mathbf{u}}_g$  are the OLS residuals and we assume  $G \rightarrow \infty$ . This yields the cluster-robust VCE given in the fourth row of table 6.1. Heteroskedasticity is just the special case where there is one observation per cluster.

A similar argument leads to the cluster-robust VCE for FGLS, where we need to restrict attention to FGLS estimators with  $\hat{\Omega} = \text{Diag}(\hat{\Omega}_g)$  so that FGLS

estimation preserves independence across clusters and  $\widehat{\mathbf{u}}_g$  are the FGLS residuals.

#### 6.2.4 Limitations of cluster–robust inference

The practitioner should be aware of several limitations of cluster–robust inference: 1) it is not always clear whether there is a need to cluster; 2) theory assumes that there are many clusters; and 3) the cluster–robust VCE can be rank deficient.

There is no standard formal test for whether there is a need to obtain cluster–robust standard errors, rather than heteroskedastic–robust standard errors. Instead, one simply uses the `vce(cluster clustvar)` option if we are in a setting where there is likely to be intracluster correlation of both the regressor and the error. If there is a meaningful difference between the heteroskedastic–robust standard errors and cluster–robust standard errors, then cluster–robust standard errors should be used following OLS estimation, and efficiency gains may be possible by GLS estimation.

Cluster–robust standard errors are based on the assumption that the number of clusters  $G$  is large. If the number of clusters is small, then cluster–robust inference leads to underestimation of standard errors, confidence intervals that are too narrow, and hypothesis tests that overreject. Studies suggest that  $G > 30$  is desirable and that even larger  $G$  may be needed if the cluster sizes are unbalanced. When  $G$  is small, one should at the very least base inference on critical values from the  $t(G - 1)$ , rather than the standard normal distribution. The `regress` command uses the  $t(G - 1)$  distribution, but many other commands, including `xtreg`, instead use standard normal critical values. Improved inference with few clusters is an active area of research; see section [6.4.6](#) for further discussion and the wild cluster bootstrap in section [12.6](#).

Finally, the cluster–robust VCE has rank at most the minimum of  $K$  (the dimension of  $\beta$ ) and  $G - 1$ . Thus, if  $K > G - 1$ , it is possible to test at most  $G - 1$  restrictions. In that case, the usual test of joint statistical significance of all regressors is not possible, and computer output will report the joint test statistic as missing. This is not a cause for concern. Tests of

fewer restrictions, including the usual tests of significance, and confidence intervals remain valid.

### 6.2.5 Leading examples of FGLS

The GLS framework is relevant whenever  $\Omega \neq \sigma^2 \mathbf{I}$ . We summarize several leading cases.

Heteroskedastic errors have already been discussed at some length and can arise in many different ways. In particular, they may reflect specification errors associated with the functional form of the model. Examples include neglected random or systematic parameter variation, incorrect functional form of the conditional mean, incorrect scaling of the variables in the regression, and incorrect distributional assumptions regarding the dependent variables. A proper treatment of the problem of heteroskedasticity may therefore require analysis of the functional form of the regression. For example, in chapter 3, a log-linear model was found to be more appropriate than a linear model.

Another common example is that of clustered (or grouped) errors, with errors being correlated within clusters but uncorrelated between clusters. A cluster consists of a group of observations that share some geographical, social, or economic trait that induces within-cluster dependence between observations, even after controlling for sources of observable differences. Such dependence can also be induced by other latent factors such as shared social norms, habits, or influence of a common local environment. In this case,  $\Omega$  can be partitioned by cluster. If all observations can be partitioned into  $G$  mutually exclusive and exhaustive groups, then  $\Omega$  can be partitioned into a block-diagonal matrix with  $G$  submatrices, simplifying the necessary inversion of the potentially very large  $N \times N$  matrix  $\Omega$ . The cluster-robust VCE can be obtained because the partitioning of errors is preserved after inversion of the block-diagonal matrix  $\Omega$ . A leading example is the RE estimator; see section 6.5.

For multivariate linear regression, such as the estimation of systems of equations, errors can be correlated across the equations for a specific individual. In this case, the model consists of  $m$  linear regression equations

$y_{ji} = \mathbf{x}'_{ji}\boldsymbol{\beta}_j + u_{ji}$ ,  $j = 1, \dots, m$ , where the errors  $u_{ji}$  are correlated over  $j$  for a given  $i$  but are uncorrelated over  $i$ . Then GLS estimation refers to efficient joint estimation of all  $m$  regressions. The three-stage least-squares estimator is an extension to the case of simultaneous-equations systems.

## 6.3 Modeling heteroskedastic data

Heteroskedastic errors are pervasive in microeconomics. The failure of the assumption of homoskedasticity in the standard regression model leads to the OLS estimator being inefficient, though it is still a consistent estimator. Given heteroskedastic errors, there are two leading approaches.

The first approach, taken in section 3.4.5, is to obtain robust estimates of the standard errors of regression coefficients without assumptions about the functional form of heteroskedasticity. Under this option, the form of heteroskedasticity has no interest for the investigator who wants to report only correct standard errors,  $t$  statistics, and  $p$ -values. This approach is easily implemented in Stata, using the `vce(robust)` option, and is by far the most common approach taken in applied economics studies.

The second approach seeks to model the heteroskedasticity and to obtain more efficient FGLS estimates. This enables more precise estimation of parameters and marginal effects and more precise prediction of the conditional mean. In practice, such gains in efficiency are often small, and OLS is commonly used.

Unlike some other standard settings for FGLS, there is no direct Stata command for FGLS estimation given heteroskedastic errors. However, it is straightforward to obtain the FGLS estimator manually, as we now demonstrate.

### 6.3.1 Simulated dataset

We use a simulated dataset, one where the conditional mean of  $y$  depends on regressors  $x_2$  and  $x_3$ , while the conditional variance depends only on  $x_2$ . The specific data-generating process (DGP) is

$$y = 1 + 1 \times x_2 + 1 \times x_3 + u; \quad x_2, x_3 \sim N(0, 25)$$
$$u = \sqrt{\exp(-1 + 0.2 \times x_2)} \times \varepsilon; \quad \varepsilon \sim N(0, 25)$$

Then the error  $u$  is heteroskedastic with a conditional variance of  $25 \times \exp(-1 + 0.2 \times x_2)$  that varies across observations according to the value taken by  $x_2$ .

We generate a sample of size 500 from this DGP:

```
. * Generated data for heteroskedasticity example
. set seed 10101
. qui set obs 500
. generate double x2 = 5*rnorinal(0)
. generate double x3 = 5*rnorinal(0)
. generate double e = 5*rnorinal(0)
. generate double u = sqrt(exp(-1+0.2*x2))*e
. generate double y = 1 + 1*x2 + 1*x3 + u
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
x2	500	-.0117399	5.073625	-16.30832	15.46858
x3	500	.0534507	5.061389	-20.59562	12.92513
e	500	.2355246	5.131809	-15.90812	12.4746
u	500	.1644912	4.234811	-22.13982	24.04547
y	500	1.206202	8.475936	-24.92206	41.67478

The generated independent and identically distributed (i.i.d.) normal variables  $x_2$ ,  $x_3$ , and, to a lesser extent,  $e$  have, approximately, means of 0 and standard deviations of 5 as expected.

### 6.3.2 OLS estimation

OLS regression with default standard errors yields

. * OLS regression with default standard errors						
. regress y x2 x3						
Source	SS	df	MS	Number of obs	=	500
Model	26914.9701	2	13457.485	F(2, 497)	=	748.65
Residual	8933.93256	497	17.9757194	Prob > F	=	0.0000
Total	35848.9026	499	71.8414882	R-squared	=	0.7508
				Adj R-squared	=	0.7498
				Root MSE	=	4.2398
y	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
x2	1.033346	.0374193	27.62	0.000	.959827	1.106866
x3	.9919595	.0375098	26.45	0.000	.9182623	1.065657
_cons	1.165312	.1896199	6.15	0.000	.792757	1.537868

The coefficient estimates are close to their true values, and the 95% confidence intervals include the DGP values. The estimates are quite precise because there are 500 observations, and for this generated dataset, the  $R^2 = 0.75$  is very high.

The standard procedure is to obtain heteroskedasticity-robust standard errors for the same OLS estimators. We have

. * OLS regression with heteroskedasticity-robust standard errors						
. regress y x2 x3, vce(robust)						
Linear regression				Number of obs	=	500
				F(2, 497)	=	501.02
				Prob > F	=	0.0000
				R-squared	=	0.7508
				Root MSE	=	4.2398
y	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]	
x2	1.033346	.0625617	16.52	0.000	.9104285	1.156265
x3	.9919595	.0342737	28.94	0.000	.9246203	1.059299
_cons	1.165312	.1899729	6.13	0.000	.7920633	1.538561

In general, failure to control for heteroskedasticity leads to default standard errors being wrong, though a priori it is not known whether they will be too large or too small. In our example, we expect the standard errors for the coefficient of  $x_2$  to be most effected because the heteroskedasticity depends on  $x_2$ . This is indeed the case. For  $x_2$ , the robust standard error is 70% higher than the incorrect default (0.063 versus 0.037). The original failure to

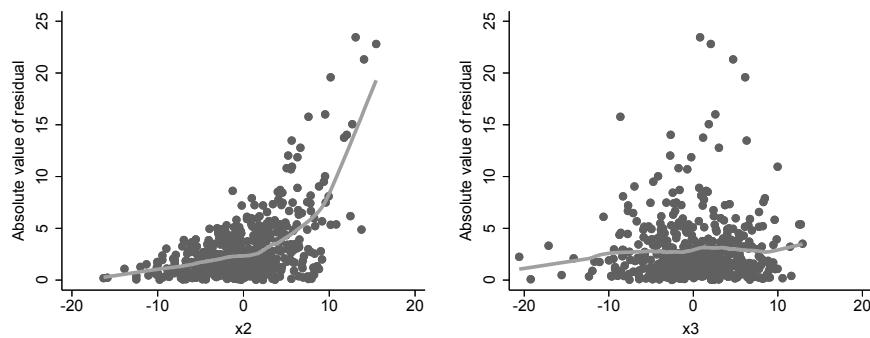
control for heteroskedasticity led to wrong standard errors, in this case, considerable understatement of the standard error of  $x_2$ . For  $x_3$ , there is much less change in the standard error.

### 6.3.3 Detecting heteroskedasticity

A simple informal diagnostic procedure is to plot the absolute value of the fitted regression residual,  $|\hat{u}_i|$ , against a variable assumed to be in the skedasticity function. The regressors in the model are natural candidates.

The following code produces separate plots of  $|\hat{u}_i|$  against  $x_{2i}$  and  $|\hat{u}_i|$  against  $x_{3i}$  and then combines these into one figure (shown in figure 6.1) by using the `graph combine` command; see section 2.6. Several options for the `twoway` command are used to improve the legibility of the graph.

```
. * Heteroskedasticity diagnostic scatterplot
. qui regress y x2 x3
. predict double uhat, resid
. generate double absu = abs(uhat)
. qui twoway (scatter absu x2) (lowess absu x2, bw(0.4) lw(thick)),
>     legend(off) ytitle("Absolute value of residual") yscale(titleg(*5))
>     xscale(titleg(*5)) plotr(style(none)) name(gls1, replace)
. qui twoway (scatter absu x3) (lowess absu x3, bw(0.4) lw(thick)),
>     legend(off) ytitle("Absolute value of residual") yscale(titleg(*5))
>     xscale(titleg(*5)) plotr(style(none)) name(gls2, replace)
. graph combine gls1 gls2, iscale(1.2) ysize(2.5) xsize(6.0)
```



**Figure 6.1.** Absolute residuals graphed against  $x_2$  and  $x_3$

It is easy to see that the range of the scatterplot becomes wider as  $x_2$  increases, with a nonlinear relationship, and is unchanging as  $x_3$  increases. These observations are to be expected given the DGP.

We can go beyond a visual representation of heteroskedasticity by formally testing the null hypothesis of homoskedasticity against the alternative that residual variances depend upon a)  $x_2$  only, b)  $x_3$  only, and c)  $x_2$  and  $x_3$  jointly. Given the previous plot (and our knowledge of the DGP), we expect the first test and the third test to reject homoskedasticity, while the second test should not reject homoskedasticity.

These tests can be implemented using the postestimation command `estat hettest`, introduced in section [3.7.4](#). It is simplest to use the `mtest` option, which performs multiple tests that separately test each component and then test all components. We have

```
. * Test heteroskedasticity depending on x2, x3, and x2 and x3
. estat hettest x2 x3, mtest
```

Breusch-Pagan/Cook-Weisberg test for heteroskedasticity

Assumption: Normal error terms

H0: Constant variance

Variable	chi2	df	p
x2	390.13	1	0.0000*
x3	3.93	1	0.0474*
Simultaneous	392.43	2	0.0000

\* Unadjusted p-values

The  $p$ -value for  $x_2$  is 0.000, causing us to reject the null hypothesis that the skedasticity function does not depend on  $x_2$ . We conclude that there is heteroskedasticity due to  $x_2$ . The  $p$ -value for  $x_3$  is 0.0474, so at level 0.05 we just reject the null hypothesis that the skedasticity function does not depend on  $x_3$ , which is not the correct decision in this case. The  $p$ -value of 0.000 for the joint (simultaneous) hypothesis leads us to conclude that the skedasticity function depends on at least one of  $x_2$  and  $x_3$ .

The `mtest` option is especially convenient if there are many regressors and, hence, many candidates for causing heteroskedasticity. It does,

however, use the version of `estat hettest` that assumes that errors are normally distributed. To relax this assumption to one of independent and identically distributed errors, we need to use the `iid` option (see section [3.7.4](#)) and conduct separate tests. Doing this leads to test statistics (not reported) with values lower than those obtained above, and the null hypothesis of no heteroskedasticity is rejected for  $x_2$  and not rejected for  $x_3$ .

### 6.3.4 FGLS estimation with heteroskedastic errors

For potential gains in efficiency, we can estimate the parameters of the model by using the two-step FGLS estimation method presented in section [6.2.2](#). For heteroskedasticity, this is easy: from [\(6.2\)](#), we need to 1) estimate  $\hat{\sigma}_i^2$  and 2) OLS regress  $y_i/\hat{\sigma}_i$  on  $x_i/\hat{\sigma}_i$ .

At the first step, we estimate the linear regression by OLS, save the residuals  $\hat{u}_i = y - \mathbf{x}'\hat{\beta}_{OLS}$ , estimate the skedasticity function  $\sigma^2(\mathbf{z}_i, \gamma)$  by regressing  $\hat{u}_i^2$  on  $\sigma^2(\mathbf{z}_i, \gamma)$ , and get the predicted values  $\hat{\sigma}^2(\mathbf{z}_i, \hat{\gamma})$ . Here our tests suggest that the skedasticity function should include only  $x_2$ . We specify the skedasticity function  $\sigma^2(\mathbf{z}) = \exp(\gamma_1 + \gamma_2 x_2)$  because taking the exponential ensures a positive variance. This is a nonlinear model that needs to be estimated by nonlinear least squares. We use the `n1` command, which is explained in section [13.3.6](#).

The first step of FGLS yields

```

. * FGLS: First step get estimate of skedasticity function
. drop uhat
. qui regress y x2 x3                                     // Get bols
. predict double uhat, resid
. generate double uhatsq = uhat^2                         // Get squared residual
. nl (uhatsq = exp({xb: x2}+{b0})), nolog    // NLS of uhatsq on exp(z`a)

```

Source	SS	df	MS	Number of obs	=	500
Model	890038.92	2	445019.458	R-squared	=	0.5975
Residual	599508.9	498	1203.83313	Adj R-squared	=	0.5959
Total	1489547.8	500	2979.09563	Root MSE	=	34.6963
				Res. dev.	=	4963.568

uhatsq	Coefficient	Std. err.	t	P> t	[95% conf. interval]
/xb_x2	.3002613	.012056	24.91	0.000	.2765743 .3239482
/b0	1.586178	.1561775	10.16	0.000	1.27933 1.893026

```

. predict double varu, yhat                                // Get sigmahat^2

```

Note that  $x_2$  explains a good deal of the heteroskedasticity ( $R^2 = 0.60$ ) and is highly statistically significant. For our DGP,  $\sigma^2(\mathbf{z}) = 25 \times \exp(-1 + 0.2x_2) = \exp(\ln 25 - 1 + 0.2x_2) = \exp(2.22 + 0.2x_2)$ , and the estimates are 1.59 and 0.30.

At the second step, the predictions  $\hat{\sigma}^2(\mathbf{z})$  define the weights that are used to obtain the FGLS estimator. Specifically, we regress  $y_i/\hat{\sigma}_i$  on  $\mathbf{x}_i/\hat{\sigma}_i$ , where  $\hat{\sigma}_i^2 = \exp(\hat{\gamma}_1 + \hat{\gamma}_2 x_{2i})$ . This weighting can be done automatically by using `aweights` in estimation. If the `aweight` is  $w_i$ , then OLS regression is of  $\sqrt{w_i}y_i$  on  $\sqrt{w_i}\mathbf{x}_i$ . Here we want the `aweight` to be  $1/\hat{\sigma}_i^2$ , or  $1/varu$ . Then,

```
. * FGLS: Second step get estimate of skedasticity function
. regress y x2 x3 [aweight=1/varu]
(sum of wgt is 338.0575221018863)
```

Source	SS	df	MS	Number of obs	=	500
Model	17266.2105	2	8633.10525	F(2, 497)	=	2530.65
Residual	1695.47363	497	3.41141576	Prob > F	=	0.0000
Total	18961.6841	499	37.999367	R-squared	=	0.9106
				Adj R-squared	=	0.9102
				Root MSE	=	1.847
y	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
x2	1.037914	.0171689	60.45	0.000	1.004181	1.071646
x3	1.01217	.0174524	58.00	0.000	.9778808	1.04646
_cons	1.348954	.1548488	8.71	0.000	1.044715	1.653193

As already noted, FGLS is a special case of a two-step estimator where it is not necessary to adjust the standard errors at the second step. Comparison with previous results for OLS with the correct robust standard errors shows that the estimated confidence intervals are narrower for FGLS. For example, for  $x_2$  the improvement is from  $[0.91, 1.16]$  to  $[1.00, 1.07]$ . As predicted by theory, FGLS with a correctly specified model for heteroskedasticity is more efficient than OLS.

The `hetregress` command with option `twostep` implements a similar WLS method, except that the weights are the exponential of the fitted value obtained from OLS regression of  $\ln(\hat{u}_i^2)$  on  $\mathbf{z}_i$ .

In practice, the form of heteroskedasticity is not known. Then a similar favorable outcome may not occur, and we should create more robust standard errors as we next consider.

### 6.3.5 Heteroskedastic-robust standard errors for the WLS estimator

The FGLS standard errors are based on the assumption of a correct model for heteroskedasticity. To guard against misspecification of this model, we use the WLS estimator presented in section [6.2.3](#), which is equal to the FGLS estimator but uses robust standard errors that do not rely on a model for heteroskedasticity. We have

```
. * WLS estimator is FGLS with robust estimate of VCE
. regress y x2 x3 [aweight=1/varu], vce(robust)
(sum of wgt is 338.0575221018863)
```

Linear regression	Number of obs = 500
	F(2, 497) = 2048.50
	Prob > F = 0.0000
	R-squared = 0.9106
	Root MSE = 1.847

y	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
x2	1.037914	.0201851	51.42	0.000	.9982548	1.077572
x3	1.01217	.0201042	50.35	0.000	.9726707	1.05167
_cons	1.348954	.1654809	8.15	0.000	1.023826	1.674083

The robust standard errors for FGLS are somewhat similar to the default standard errors for FGLS, as expected because here FGLS is known to use the DGP model for heteroskedasticity. They are substantially smaller than the robust standard errors for OLS.

## 6.4 OLS for clustered data

Cluster-robust inference for OLS was introduced in section [3.4.6](#). Here we provide greater detail.

Before we proceed, however, note that the topic of inference with clustered errors should not be confused with the quite different topic of cluster analysis, identifying groupings within the data, which can be implemented using the `cluster` command.

We consider OLS estimation for data on individual  $i$  in cluster  $g$  (of  $G$  clusters), so

$$y_{gi} = \mathbf{x}'_{gi}\boldsymbol{\beta} + u_{gi}, \quad i = 1, \dots, N_g, \quad g = 1, \dots, G$$

where clustering is on the first subscript  $g$ . Note that we could instead use the alternative ordering of the subscripts  $y_{ig}$ . We use  $y_{gi}$  here to be consistent with short panel data  $y_{it}$  for which clustering is often on the individual, the first subscript.

### 6.4.1 Clustered dataset

We consider data on use of medical services by individuals, where individuals are clustered within household and, additionally, households are clustered in villages (called communes in this dataset). The data, from Vietnam, are the same as those used in [Cameron and Trivedi \(2005, 852\)](#).

Before analysis, we drop one household whose members were split between two communes and drop one observation with missing data. We have

```

. * Read in Vietnam clustered data, and delete one household in two communes
. qui use mus206vlss, clear
. drop if lnhhexp > 2.579681 & lnhhexp < 2.579683
(12 observations deleted)
. drop if missing(lnhhexp)
(1 observation deleted)

```

The dependent variable is the number of direct pharmacy visits (`pharvis`). The independent variables are the logarithm of household medical expenditures (`lnhhexp`) and the number of illnesses (`illness`). The data cover 12 months.

The `commune` variable identifies the 194 separate villages. However, the dataset does not include a household identifier. We create the household identifier `hh` using the fact that the `lnhhexp` variable takes on a distinct value for each household. We have

```

. * Create a unique household identifier and summarize data
. egen hh = group(lnhhexp)
. summarize pharvis lnhhexp illness hh commune

```

Variable	Obs	Mean	Std. dev.	Min	Max
pharvis	27,753	.5110439	1.312606	0	30
lnhhexp	27,753	2.60262	.6245493	.0467014	5.405502
illness	27,753	.6218427	.8995018	0	9
hh	27,753	3097.765	1601.653	1	5739
commune	27,753	101.514	56.27264	1	194

There are 5,739 households.

The `pharvis` variable is a count that is best modeled by using count regression commands such as `poisson` and `xtpoisson`; this is done in section [13.9](#). In this chapter, we use the same data to illustrate linear regression with clustered data.

#### 6.4.2 OLS and cluster–robust standard errors

One complication of clustering is that the error is correlated within cluster. If that is the only complication, then valid inference simply uses standard cross-sectional estimators along with cluster–robust standard errors that are

given in table 6.1. The overall  $F$  statistic and individual  $t$  statistics and confidence intervals use  $G - 1$  degrees of freedom.

```
. * OLS estimation with cluster--robust standard errors clustering on household
. regress pharvis lnhhexp illness, vce(cluster hh)

Linear regression
Number of obs      =    27,753
F(2, 5738)        =   579.82
Prob > F          =  0.0000
R-squared          =  0.1817
Root MSE           =  1.1874

(Std. err. adjusted for 5,739 clusters in hh)
```

pharvis	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
lnhhexp	.0247159	.0139563	1.77	0.077	-.0026437	.0520754
illness	.6235444	.0183113	34.05	0.000	.5876474	.6594414
_cons	.0589713	.03666	1.61	0.108	-.0128961	.1308387

One more `illness` is associated with 0.624 more pharmacy visits, a very large effect. By contrast, an increase in household medical expenditures has a very small effect because a one-unit change in the natural logarithm of household medical expenditures, a very large change, is associated with only 0.025 more pharmacy visits.

Here we contrast the different standard errors obtained with no correction for clustering, clustering on household, and clustering on village. We have

```
. * OLS estimation with various cluster--robust standard errors
. qui regress pharvis lnhhexp illness
. estimates store OLS_iid
. qui regress pharvis lnhhexp illness, vce(robust)
. estimates store OLS_het
. qui regress pharvis lnhhexp illness, vce(cluster hh)
. estimates store OLS_hh
. qui regress pharvis lnhhexp illness,
> vce(bootstrap, cluster(hh) seed(10101) reps(400))
. estimates store OLS_boot
. qui regress pharvis lnhhexp illness, vce(cluster commune)
. estimates store OLS_comm
```

```
. estimates table OLS_iid OLS_het OLS_hh OLS_boot OLS_comm,
>     b(%10.4f) se stats(r2 N)
```

Variable	OLS_iid	OLS_het	OLS_hh	OLS_boot	OLS_comm
lnhhexp	0.0247	0.0247	0.0247	0.0247	0.0247
	0.0115	0.0109	0.0140	0.0135	0.0211
illness	0.6235	0.6235	0.6235	0.6235	0.6235
	0.0080	0.0141	0.0183	0.0177	0.0342
_cons	0.0590	0.0590	0.0590	0.0590	0.0590
	0.0316	0.0292	0.0367	0.0361	0.0556
r2	0.1817	0.1817	0.1817	0.1817	0.1817
N	27753	27753	27753	27753	27753

Legend: b/se

The effect of correction of standard errors for heteroskedasticity is unknown a priori. Here there is little effect on the standard errors for the intercept and `lnhhexp`, though there is a large increase for `illness`.

More importantly, controlling for clustering is expected to increase reported standard errors, especially for regressors highly correlated within the cluster. Here standard errors increase by around 30% as we move to clustering on household and by another 50% or more as we move to clustering on village. In total, clustering on village leads to a doubling of standard errors compared with heteroskedastic–robust standard errors.

The fourth column provides cluster-pairs bootstrap standard errors based on 400 bootstrap replications, where the bootstrap resampling is over households. These are close (within 4%) to the cluster-robust standard errors given in the third column, as expected given the asymptotic equivalence of the two methods.

#### 6.4.3 Intracluster correlation and inflation of OLS standard errors

The impact of clustering on default OLS standard errors depends in part on the intracluster correlation of regressors and the intracluster correlation of model errors. The intracluster correlation coefficient can be obtained using the `loneway` command.

```

. * Within commune intracluster correlation for lnhhexp
. loneway lnhhexp commune

      One-way analysis of variance for lnhhexp: log of total household expenditure

                           Number of obs =      27,753
                                         R-squared =    0.5014

Source          SS           df        MS         F       Prob > F
-----+-----+-----+-----+-----+-----+
Between commune   5427.1435    193    28.119914    143.57    0.0000
Within commune    5397.8513  27,559    .19586528
-----+-----+
Total            10824.995  27,752    .39006179
-----+-----+
Intraclass      Asy.
correlation      S.E.      [95% conf. interval]
-----+-----+
          0.49920    0.02621    0.44782    0.55057
Estimated SD of commune effect      .4418549
Estimated SD within commune      .4425667
Est. reliability of a commune mean      0.99303
(evaluated at n=143.03)

```

For variable `lnhhexp`, the within-commune intracluster correlation is 0.499.

Cluster-robust standard errors are approximately  $\tau = \sqrt{1 + \rho_x \rho_u (M - 1)}$  times the default OLS standard errors, where  $\rho_x$  and  $\rho_u$  are intracluster correlations of  $x$  and  $u$  and  $M$  is the average number of observations per cluster (see section [3.4.6](#)). The following code does this computation for the standard error of the estimated coefficient of `lnhhexp` when clustering is on `commune`. We have

```

. * Standard-error inflation factor for b_lnhhexp when clustering on commune
. qui loneway lnhhexp commune
. scalar rho_x = r(rho)
. qui regress pharvis lnhhexp illness
. scalar numobs = e(N)
. predict uhat, resid
. qui loneway uhat commune
. scalar rho_uhat = r(rho)
. qui sum
. scalar M = numobs/194
. display "rho_x = " rho_x " rho_u = " rho_uhat " M = " M
>      _n "Standard error inflation = " sqrt(1+rho_x*rho_uhat*(M-1))
rho_x = .49919517 rho_u = .04866825 M = 143.0567
Standard error inflation = 2.1098013

```

This calculation of standard error inflation of 2.11 is only a guide. In fact, the standard error increased from 0.0115 to 0.0211, an inflation factor of  $0.0211/0.0115 = 1.83$ . But the computation does provide some intuition. While the within-commune correlation of residuals was low (0.049), there were many individuals per cluster (143 on average) and high within-commune correlation of `lnhhexp`.

A variation mentioned in section [3.4.6](#) uses  $\tau = \sqrt{1 + \rho_{xu}(M - 1)}$ , where  $\rho_{xu}$  is the within-cluster correlation of  $x_{gi}u_{gi}$ . This yields estimated standard error inflation of 2.68.

#### 6.4.4 Two-way cluster–robust standard errors for OLS

Two-way clustering arises when clustered errors arise for two distinct reasons that are nonnested, such as for matched employer–employee data with potentially multiple observations for each employer and each employee.

Let  $\widehat{V}_1$  denote the cluster–robust variance VCE obtained by clustering on the first dimension, say, employer;  $\widehat{V}_2$  denote the estimate obtained by clustering on the second dimension, say, employee; and  $\widehat{V}_{12}$  denote the estimate obtained by clustering on the intersection of the two ways, say, employer–employee pair. Then the two-way cluster–robust estimator of the VCE of the OLS estimator is

$$\widehat{V}_{\text{2way}} = \widehat{V}_1 + \widehat{V}_2 - \widehat{V}_{12} \quad (6.3)$$

where the subtraction of  $\widehat{V}_{12}$  corrects for overcounting of observations that were clustered in both dimensions. This variance estimator, proposed independently by [Cameron, Gelbach, and Miller \(2011\)](#), [Thompson \(2011\)](#), and [Miglioretti and Heagarty \(2006\)](#), relies on asymptotic theory that the number of clusters in each dimension goes to infinity.

The community-contributed command `vcemway` ([Gu and Yoo 2019](#)) implements the two-way variance estimate not only for OLS but for any estimation command that stores in `e(V)` a one-way cluster–robust variance

matrix estimate. The default follows the community-contributed `cmsgreg` command ([Cameron, Gelbach, and Miller 2011](#)) and uses different numbers of clusters in computing the degrees-of-freedom adjustments for the three variances in (6.3). The `vcemway` option `vmcfactor(minimum)` instead uses  $G_{\min} = \min(G, H)$  in computing each of  $\widehat{V}_1$ ,  $\widehat{V}_2$ , and  $\widehat{V}_3$ , as do the `ivreg2` command ([Baum, Schaffer, and Stillman 2007](#)) with option `small` (presented in section [7.6.9](#)) and the community-contributed command `reg2hdfe` ([Guimarães and Portugal 2010](#)).

The two-way cluster–robust VCE is not guaranteed to be positive definite, because while each of the three component matrices in (6.3) is positive definite, the subtraction of the third matrix can lead to a VCE that is not positive definite. Then standard errors cannot be computed. The `vcemway` command follows the `cmsgreg` command by setting any negative eigenvalues of  $\widehat{V}_{2\text{way}}$  to zero and then reconstructing the VCE using an eigenvalue decomposition of the matrix. A conservative alternative instead drops the third term in (6.3).

In the current application, there is no obvious reason for clustering on a second dimension. For purely illustrative purposes, we cluster in two nonnested dimensions: `commune` and `illdays`. Using the `vcemway` command, we obtain

```

. * Two-way cluster--robust standard errors using vcemway command
. vcemway regress pharvis lnhhexp illness, cluster(commune illdays)
Linear regression
Number of obs      =    27,753
F(2, 193)          =   171.28
Prob > F          =  0.0000
R-squared           =  0.1817
Root MSE            =  1.1874
(Std. err. adjusted for clustering on commune illdays)

```

pharvis	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
lnhhexp	.0247159	.0312952	0.79	0.436	-.039111	.0885428
illness	.6235444	.0661737	9.42	0.000	.4885824	.7585065
_cons	.0589713	.061069	0.97	0.342	-.0655797	.1835223

Notes:

Std. Err. adjusted for 2-way clustering on commune illdays

Number of clusters in commune = 194

Number of clusters in illdays = 32

Stata's default small-cluster correction factors have been applied. See  
> vcemway for detail.

Residual degrees of freedom for t tests and F tests = 31

F(,) and Prob > F above only account for one-way clustering on commune.

Use test to compute F(,) and Prob > F that account for 2-way clustering.

The two-way cluster-robust standard errors are larger than the earlier one-way cluster-robust standard errors with clustering on commune. The  $p$ -values and confidence intervals are based on the  $t(31)$  distribution because here  $\min(G, H) = 32$ . For estimation commands such as logit that use the standard normal distribution, it is better to use the vcemway option vmdfr(#), where # equals  $\min(G, H) - 1$ .

Finally, we compare separate one-way clustering on commune or on illdays with two-way clustering on both variables. We have

```

. * Effect of two-way versus separate one-way clustering
. qui regress pharvis lnhhexp illness, vce(robust)
. estimates store het
. qui regress pharvis lnhhexp illness, cluster(commune)
. estimates store commune
. qui regress pharvis lnhhexp illness, cluster(illdays)
. estimates store illdays
. qui vcemway regress pharvis lnhhexp illness, cluster(commune illdays)
. estimates store twoway
. qui vcemway regress pharvis lnhhexp illness, cluster(commune illdays)
>      vmcfactor(minimum)
. estimates store twowayalt

. estimates table het commune illdays twoway twowayalt, b(%10.4f) se stats(r2 N)

```

Variable	het	commune	illdays	twoway	twowayalt
lnhhexp	0.0247	0.0247	0.0247	0.0247	0.0247
	0.0109	0.0211	0.0271	0.0313	0.0314
illness	0.6235	0.6235	0.6235	0.6235	0.6235
	0.0141	0.0342	0.0599	0.0662	0.0663
_cons	0.0590	0.0590	0.0590	0.0590	0.0590
	0.0292	0.0556	0.0448	0.0611	0.0614
r2	0.1817	0.1817	0.1817	0.1817	0.1817
N	27753	27753	27753	27753	27753

Legend: b/se

The standard errors with two-way clustering exceed those with either separate one-way clustering.

#### 6.4.5 Dyad-robust standard errors for OLS

Dyadic data are data with dependent variable  $y_{ij}$  that measures the relationship between individual  $i$  and  $j$ . Examples include friendship interactions, interhousehold interactions, and trade flows between countries.

For such data, it is possible that a model error  $u_{ij}$  will be correlated with every other model error  $u_{kl}$  if any of the two components of the pair  $(k, l)$  is common with any of the two components of the pair  $(i, j)$ . Two-way cluster-robust standard errors control for error correlation when  $k = i$  or

$l = j$ , or both. Dyadic-robust standard errors additionally control for error correlation when  $k = j$  or  $l = i$ , or both.

For the formula for dyadic-robust standard errors, see [Cameron and Miller \(2014\)](#) or [Aranow, Samii, and Assenova \(2015\)](#), and for asymptotic theory, see [Tabord-Meehan \(2019\)](#).

The need to control for dyadic correlation is greater the more links there are across observations. [Fafchamps and Gubert \(2007\)](#) proposed this correction for social networks across households, where households on average interacted with only a few other households, and found that the dyadic correction made little difference to standard errors.

By contrast, [Cameron and Miller \(2014\)](#) considered data on trade flows between many countries and found that dyadic-robust standard errors could be several times standard errors that do not make any correction (and are about 25% larger than two-way cluster-robust standard errors). Furthermore, if panel data  $y_{ij,t}$  are available and dyad-specific fixed effects are included, there is still great need to control for dyadic error correlation.

#### 6.4.6 Cluster-robust inference with few clusters

The underlying asymptotic theory assumes that the number of clusters  $G \rightarrow \infty$ . It is not unusual to have applications where there are few clusters, such as when clustering is on state or province. In that case, the usual asymptotic theory can lead to tests with considerable size distortion, typically overrejecting, and confidence intervals that typically undercover.

A similar situation would arise in a cross-sectional setting with, say,  $N = 20$  independent observations, but with so few observations, estimation is then often so imprecise that there is no reason to go further and use improved inferential methods. By contrast, it is not unusual to obtain reasonably precise estimates with  $G = 10$  or even lower, given many observations per cluster, so there is then a reason to improve on the usual asymptotic methods.

A simple correction is to base inference on the  $t$  and  $F$  distributions rather than the standard normal and chi-squared distributions. This can be

done after any estimation command using the `test` command or `testparm` command with option `df(#)`; a common choice is to use degrees of freedom  $G - 1$ . A few Stata linear estimation commands, notably `regress`, automatically provide output with tests and confidence intervals for coefficients based on the  $t(G - 1)$  and  $F(q, G - 1)$  distributions. And the `ivregress` with option `small` does this. But especially nonlinear commands such as `poisson` provide output based only on standard normal and chi-squared distributions. The option `df(#)` of the `test` command following any estimation command uses the  $F(9, \#)$  distribution.

Suppose  $G = 10$ . If a Wald test has  $t = 1.96$ , then using the  $t(G - 1)$  distribution leads to a two-sided  $p$ -value of 0.082, computed as `2*ttail(9, 1.96)`, rather than 0.05 if the standard normal distribution was used. Even this correction is not enough in practice, and simulation studies find that the effective degrees of freedom can be much less than  $G - 1$ . This is especially the case if the clusters are unbalanced because of differences in cluster sizes or substantial differences in regressor values across clusters. [Carter, Schnepel, and Steigerwald \(2017\)](#) present results, and the associated community-contributed `clusteff` command ([Lee and Steigerwald 2018](#)), illustrated in section 12.6.3, provides a conservative estimate of the effective number of clusters.

With unbalanced clusters, some clusters can greatly influence parameter estimates and also lead to test size distortion. [MacKinnon, Nielsen, and Webb \(Forthcoming\)](#) provide a discussion, and the community-contributed `summcluster` package ([MacKinnon, Nielsen, and Webb 2022](#)) provides measures of high-leverage clusters and influential clusters.

And with few clusters the VCE should use a finite degrees of freedom correction such as Stata's scaling by  $G/(G - 1)$  or, for linear regression,  $N/(N - K)$  times  $G/(G - 1)$ .

Some specialized methods that are oriented to specific settings of few clusters have been proposed; Cameron and Miller ([2015](#)) and MacKinnon, Nielsen, and Webb ([Forthcoming](#)) provide reviews. These include settings with many observations per clusters and settings with just one treated cluster. If the regressor of interest varies within cluster, [Ibragimov and Müller \(2010\)](#) propose a test that is conservative (provided the test level

$\alpha \leq 0.083$ ) based on separate OLS estimation in each cluster. If the regressor of interest is the same for all units in the cluster, which is often the case in a treatment evaluation setting, then [Donald and Lang \(2007\)](#) propose use of a two-step estimator under the assumption of homogeneous clusters.

[Hagemann \(2020\)](#) proposes a rearrangement test when there is only one treated cluster and provides many related references.

A general method for inference with few clusters applies the percentile- $t$  method to a particular bootstrap, the wild cluster bootstrap; and the community-contributed `boottest` command ([Roodman et al. 2019](#)) implements this bootstrap; see section [12.6](#). The command can be applied following a wide range of estimation commands used in analyzing clustered data, including following `regress`, `areg`, and `xtreg`, `fe`. Improved inference with few clusters is a current area of research.

## 6.5 FGLS estimators for clustered data

For cross-sectional datasets, such as data on individuals with clustering on village, there are two ways to implement FGLS with clustered data. First, one can adapt panel-data methods, using the `xtreg` commands, because panel data also have clustered errors under the standard panel assumption of uncorrelated errors across individuals but error correlation over time for a given individual. Second, one can directly use multilevel mixed-effects regression, using the `mixed` command. In the simplest case of models with an i.i.d. normally distributed random intercept, the two methods lead to the same results.

The first approach is the approach usually taken by economists and is presented in sections [6.5](#) and [6.6](#). The second approach, used in many other disciplines, is deferred to section [6.7](#).

In sections [6.5](#) and [6.6](#), we use the `xtset` command, the `xtreg` command (with options `pa`, `re`, and `fe`), and the `xtdescribe` command. These commands are presented in great detail in chapter [8](#); the presentation in the current chapter is much briefer.

### 6.5.1 Defining the cluster identifier

While we apply panel-data commands to this cross-sectional example, it is very important to note an important conceptual change from the panel-data case.

For panels, there are multiple observations per individual, so clustering is on the individual ( $i$ ). Here, instead, there are multiple observations per household or per village, so clustering is on the household or village ( $g$ ).

To use the `xt` commands, we first need to define the variable that identifies the cluster, using the `xtset` command. In the current cross-sectional setting, this is the household or village.

In the remainder of this chapter, we focus on clustering on the household. We have

```

. * Define the cluster identifier variable
. xtset hh
Panel variable: hh (unbalanced)

```

## 6.5.2 FGLS estimator for population-averaged model

The FGLS estimator requires specification of a model for within-cluster correlation of the error  $u_{gi}$  in the linear model  $y_{gi} = \mathbf{x}'_{gi}\beta + u_{gi}$ .

A natural model for data clustered on, for example, village or region or school is that the error correlation is the same for any pair of individuals in the same cluster. That is,  $\text{Cor}(u_{gi}, u_{gj}) = \rho$  for  $i \neq j$ , an assumption called one of equicorrelation or of exchangeable errors.

The consequent FGLS estimator can be obtained using the `xtreg, pa` command with option `corr(exchangeable)`. If equicorrelation is the wrong model for error correlation, the FGLS estimator of  $\beta$  remains consistent, but default standard errors are inconsistent. Instead we use cluster-robust standard errors. We obtain

```

. * FGLS with equicorrelated errors using using xtreg, pa
. xtreg pharvis lnhhexp illness, pa corr(exchangeable) vce(robust)

Iteration 1: tolerance = .01925861
Iteration 2: tolerance = .00001094
Iteration 3: tolerance = 7.437e-09

GEE population-averaged model
Group variable: hh
Family: Gaussian
Link: Identity
Correlation: exchangeable
Number of obs      = 27,753
Number of groups = 5,739
Obs per group:
               min =        1
                  avg =     4.8
                  max =     19
Wald chi2(2)      = 1346.73
Prob > chi2       = 0.0000
Scale parameter = 1.409849
(Std. err. adjusted for clustering on hh)

```

pharvis	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
lnhhexp	.0197198	.0146096	1.35	0.177	-.0089145	.0483541
illness	.6185181	.0168691	36.67	0.000	.5854552	.6515809
_cons	.0793774	.0388975	2.04	0.041	.0031398	.1556151

From supplemental output, the FGLS estimates are based on a within-cluster error correlation estimate of 0.164.

### 6.5.3 Cluster-specific random-effects model

For individual  $i$  in cluster  $g$  (of  $G$  clusters), the cluster-specific effects model is

$$y_{gi} = \mathbf{x}'_{gi}\boldsymbol{\beta} + \alpha_g + \varepsilon_{gi}, \quad i = 1, \dots, N_g, \quad g = 1, \dots, G \quad (6.4)$$

The change from the usual linear regression model is to allow the intercept  $\alpha_g$  to vary across the  $G$  clusters, which will at least partially control for within-cluster correlation.

In the RE model,  $\alpha_g$  is an i.i.d. random variable. (In the FE model presented in the subsequent section,  $\alpha_g$  is not i.i.d. and furthermore is potentially correlated with the regressors.)

More precisely, the RE model assumes that in (6.4) the cluster-specific random effect  $\alpha_g$  is i.i.d.  $(0, \sigma_\alpha^2)$  and the individual-specific effect is i.i.d.  $(0, \sigma_\varepsilon^2)$ . The additional component  $\alpha_g$  in the model error induces within-cluster error correlation. Specifically, for different individuals in the same cluster, the error correlation is  $\sigma_\alpha^2 / (\sigma_\alpha^2 + \sigma_\varepsilon^2)$ . This is an example of equicorrelation, with error correlation the same for any pair of observations in the cluster. This assumption of exchangeable errors is quite reasonable here because there is no natural ordering of household members. We expect similar results to those from the preceding population-averaged estimator based on exchangeable errors.

### 6.5.4 RE estimator

The RE FGLS estimator can be obtained using the `xtreg, re` command defined in section [8.2.4](#).

If the RE model for the errors is misspecified, then the RE estimator of  $\beta$  remains consistent, but default standard errors will be incorrect. Instead, we report cluster-robust standard errors that are valid provided  $G \rightarrow \infty$ .

```
. * Random hh effects FGLS using xtreg, re
. xtreg pharvis lnhhexp illness, re vce(robust)

Random-effects GLS regression
Group variable: hh
Number of obs      = 27,753
Number of groups  = 5,739
Obs per group:
min = 1
avg = 4.8
max = 19
Wald chi2(2)      = 1371.37
corr(u_i, X) = 0 (assumed)
Prob > chi2       = 0.0000
(Std. err. adjusted for 5,739 clusters in hh)
```

pharvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lnhhexp	.0182668	.0150469	1.21	0.225	-.0112245 .0477581
illness	.6165335	.0166859	36.95	0.000	.5838298 .6492372
_cons	.0858413	.0404213	2.12	0.034	.006617 .1650655
sigma_u	.62622406				
sigma_e	1.0606587				
rho	.25848181	(fraction of variance due to u_i)			

The slope coefficients are 0.018 and 0.617 compared with 0.025 and 0.624 for OLS. The estimates of  $\sigma_\alpha$  and  $\sigma_\varepsilon$  are, respectively, 0.626 and 1.061, so the household-specific random effect is important, leading to within-household error correlation of 0.258.

A variation of the RE estimator is to additionally assume that  $\alpha_g$  and  $\varepsilon_{gi}$  are normally distributed. Then the MLE can be obtained using the `xtreg, mle` command. From output given below, this yields very similar slope coefficients of 0.0192 and 0.6179. The MLE estimates of  $\sigma_\alpha$  and  $\sigma_\varepsilon$  are, respectively, 0.524 and 1.071.

### 6.5.5 Cluster-robust standard errors for the RE estimator

The default standard errors, based on the VCE formula given in the sixth row of table 6.1, are valid only if the model for the errors is correctly specified,

that is, if  $\alpha_g$  and  $\varepsilon_{gi}$  are i.i.d. These are strong assumptions that impose equicorrelation of errors in the same cluster and assume homoskedasticity.

We should instead use cluster-robust standard errors, based on the VCE formula given in the last row of table 6.1. This is possible because the RE estimator preserves independence across clusters, though we need to assume that the number of clusters is large.

We compare VCEs in increasing order of control for clustering: no control, household clusters, and commune clusters. We have

```
. * Random hh effects FGLS with various standard errors
. qui xtreg pharvis lnhhexp illness, re mle
. estimates store RE_MLE
. qui xtreg pharvis lnhhexp illness, re
. estimates store RE_def
. qui xtreg pharvis lnhhexp illness, re vce(robust)
. estimates store RE_rob
. qui xtreg pharvis lnhhexp illness, re vce(cluster hh)
. estimates store RE_hh
. qui xtreg pharvis lnhhexp illness, re vce(cluster commune)
. estimates store RE_comm
. estimates table RE_MLE RE_def RE_rob RE_hh RE_comm RE_hh,
>      keep(lnhhexp illness _cons) b(%10.4f) se stats(N) eq(1)
```

Variable	RE_MLE	RE_def	RE_rob	RE_hh	RE_comm
lnhhexp	0.0192	0.0183	0.0183	0.0183	0.0183
	0.0154	0.0168	0.0150	0.0150	0.0217
illness	0.6179	0.6165	0.6165	0.6165	0.6165
	0.0082	0.0083	0.0167	0.0167	0.0296
_cons	0.0816	0.0858	0.0858	0.0858	0.0858
	0.0413	0.0448	0.0404	0.0404	0.0583
N	27753	27753	27753	27753	27753

Legend: b/se

The first column gives the RE-MLE estimates (with default standard errors) that are close to the subsequent RE estimates. The default standard errors for RE estimates, given in the second column, substantially underestimate the standard error of `illness`. The third and fourth columns are identical and demonstrate that `vce(robust)` is equivalent to the `vce(cluster clustvar)`

option, provided that *clustvar* is the variable given in the `xtset` command. These are the standard errors that would most often be used. The fifth column clusters at the broader level of `commune`. This is possible because households are nested within villages and leads to even larger standard errors.

## 6.6 Fixed-effects estimator for clustered data

The FE model is less restrictive than the RE model because it allows a component of the error, one that is cluster specific, to be correlated with the regressors. This reduces the likelihood that estimators are inconsistent.

### 6.6.1 Cluster-specific FE model

The cluster-specific FE model specifies that for individual  $i$  in cluster  $j$

$$y_{gi} = \mathbf{x}'_{gi}\beta + \alpha_g + \varepsilon_{gi}, \quad i = 1, \dots, N_g, \quad g = 1, \dots, G \quad (6.5)$$

where the cluster-specific effect  $\alpha_g$  is a fixed effect, one potentially correlated with the regressors.

Like a random effect, which instead assumes that  $\alpha_g$  is purely random, this can at least partially control for within-cluster correlation.

More substantively, it relaxes the RE assumption that the cluster-specific effect is uncorrelated with the regressors. If  $\alpha_g$  is indeed correlated with  $\mathbf{x}_{gi}$ , then both the OLS and RE estimators are inconsistent. The FE estimator, presented below, is consistent provided only that  $E(\varepsilon_{gi}|\mathbf{x}_{gi}, \alpha_g) = 0$ . That is, after we control for regressors and a cluster-specific effect that may be correlated with the regressors, any remaining error is assumed to be uncorrelated with the regressors. The RE estimator requires the much stronger conditions that  $\varepsilon_{gi}$  and  $\alpha_g$  be uncorrelated with the regressors.

Thus, the FE estimator is widely used in microeconomics studies with clustered data.

### 6.6.2 FE estimator

The FE estimator of  $\beta$ , the FGLS estimator in the FE model assuming that the error  $\varepsilon_{ij}$  is i.i.d., can be shown to be equivalent to the OLS estimator in the

transformed model, called the within model or mean-differenced model,

$$(y_{gi} - \bar{y}_g) = (\mathbf{x}_{gi} - \bar{\mathbf{x}}_g)' \boldsymbol{\beta} + (\varepsilon_{gi} - \bar{\varepsilon}_g) \quad (6.6)$$

where, for example,  $\bar{\mathbf{x}}_g = N_g^{-1} \sum_{i=1}^{N_g} \mathbf{x}_{gi}$ . Note that variables that are invariant within cluster drop out because  $x_{gi} = x_g$  for all  $i$  implies  $\bar{x}_g = x_g$ . This is a regression of the within-cluster variation of  $y$  on the within-cluster variation of  $\mathbf{x}$ . Thus, the FE estimator is also called the within estimator.

The FE estimator is consistent, regardless of whether the cluster-specific effects are fixed or random. But there can be a loss of efficiency because the FE estimator uses only within-cluster variation in the data and because coefficients of cluster-invariant regressors are not identified.

There are several ways to obtain the FE estimator of  $\boldsymbol{\beta}$ . The preferred method is to use the `xtreg, fe` command, with cluster-robust standard errors.

```
. * FE estimation with hh FE using xtreg, fe
. xtreg pharvis lnhhexp illness, fe vce(robust)
note: lnhhexp omitted because of collinearity.

Fixed-effects (within) regression
Group variable: hh
R-squared:
    Within = 0.1541
    Between = 0.2069
    Overall = 0.1816
corr(u_i, Xb) = 0.0160
Number of obs      = 27,753
Number of groups  = 5,739
Obs per group:
    min = 1
    avg = 4.8
    max = 19
F(1,5738)          = 1155.53
Prob > F          = 0.0000
(Std. err. adjusted for 5,739 clusters in hh)
```

pharvis	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
lnhhexp	0	(omitted)				
illness	.6089755	.0179147	33.99	0.000	.5738559	.644095
_cons	.1323569	.0111401	11.88	0.000	.1105181	.1541957
sigma_u	.82219146					
sigma_e	1.0606587					
rho	.37534725	(fraction of variance due to u_i)				

Note that the coefficient of `lnhhexp` is not identified. This is because `lnhhexp` is the same for every individual in the household, so there is no within-cluster variation in `lnhhexp`, so  $(x_{gi} - \bar{x}_g) = 0$  for this regressor. The coefficient of `illness` is 0.609, similar to the OLS estimate of 0.624.

The estimated intercept is nonzero because the `xtreg, fe` command modifies (6.6) to instead estimate

$$(y_{gi} - \bar{y}_g + \bar{\bar{y}}) = \alpha + (\mathbf{x}_{gi} - \bar{\mathbf{x}}_g + \bar{\bar{\mathbf{x}}})' \boldsymbol{\beta} + (\varepsilon_{gi} - \bar{\varepsilon}_g + \bar{\bar{\varepsilon}}) \quad (6.7)$$

where  $\bar{\bar{y}}$  and  $\bar{\bar{\mathbf{x}}}$  are the overall means of  $y$  and  $\mathbf{x}$ . This yields the same slope estimates.

If the coefficient of `lnhhexp` is of intrinsic interest, then clearly the FE estimator is of no use in this application. But if the coefficient of `illness` is of intrinsic interest, then the FE estimator provides consistent estimates even if `illness` is correlated with the household effect  $\alpha_g$ . Consistency requires only that, after we control for other regressors and the cluster-specific effect  $\alpha_g$ , `illness` is uncorrelated with the remaining error  $\varepsilon_{gi}$ .

### 6.6.3 Cluster–robust standard errors for the FE estimator

Similar to the discussion for the RE estimator, even after inclusion of the cluster-specific fixed effects  $\alpha_g$ , there is usually remaining heteroskedasticity and within-cluster error correlation, so inference should be based on cluster–robust standard errors.

We compare VCES for the FE estimator, estimated using `xtreg, fe`, in increasing order of control for clustering: no control, household clusters, and village clusters.

```

. * FE estimation with hh RE and various standard errors
. qui xtreg pharvis lnhhexp illness, fe
. estimate store FE_def
. qui xtreg pharvis lnhhexp illness, fe vce(robust)
. estimate store FE_rob
. qui xtreg pharvis lnhhexp illness, fe vce(cluster hh)
. estimate store FE_hh
. qui xtreg pharvis lnhhexp illness, fe vce(cluster commune)
. estimate store FE_comm
. qui regress pharvis lnhhexp illness, vce(cluster hh)
. estimates store OLS_hh
. estimates table FE_def FE_rob FE_hh FE_comm OLS_hh, b(%10.4f) se stats(N)

```

Variable	FE_def	FE_rob	FE_hh	FE_comm	OLS_hh
lnhhexp	(omitted)	(omitted)	(omitted)	(omitted)	0.0247 0.0140
illness	0.6090 0.0096	0.6090 0.0179	0.6090 0.0179	0.6090 0.0276	0.6235 0.0183
_cons	0.1324 0.0087	0.1324 0.0111	0.1324 0.0111	0.1324 0.0172	0.0590 0.0367
N	27753	27753	27753	27753	27753

Legend: b/se

The default standard errors, given in the first column, substantially underestimate the standard error of `illness`. The second and third columns are identical and demonstrate that the `vce(robust)` is equivalent to the `vce(cluster clustvar)` option, provided that `clustvar` is the variable given in the `xtset` command. These are the standard errors that would most often be used. The fourth column clusters at the broader level of `commune`. This is possible because households are nested within communes and leads to even larger standard errors.

The FE estimator is often less efficient than the OLS estimator, sometimes substantially so, because it uses only within variation of the data. In this application, however, the third and fifth columns, which both cluster on household, display little difference in the standard error of `illness`.

#### 6.6.4 Alternative methods for computing the FE estimator

There are alternative methods for computing the FE estimator in the model (6.5). Even though these lead to exactly the same estimates of  $\beta$  and similar default standard errors, we strongly recommend that the `xtreg, fe` command be used. First, `xtreg, fe` (and `areg`) are computationally faster than using `regress` with an indicator variable for each individual. Second, if there are few observations per cluster, then commands other than `xtreg, fe` overstate standard errors because of the way degrees of freedom are calculated.

One approach is to directly estimate both  $\alpha_g$ ,  $g = 1, \dots, N_g$ , and  $\beta$  in the model (6.5). This is the least-squares dummy-variable (LSDV) estimator, obtained by the OLS regression

$$y_{gi} = \left( \sum_{h=1}^G \alpha_h d_{h,gi} \right) + \mathbf{x}'_{gi} \boldsymbol{\beta} + \varepsilon_{gi} \quad (6.8)$$

where there are  $G$  individual-specific indicator variables  $d_{h,gi}$ ,  $g = 1, \dots, G$ , with  $d_{h,gi} = 1$  for the  $g$ ith observation if  $h = g$  and  $d_{h,gi} = 0$  otherwise.

This brute-force method involves inversion of a  $(G + K) \times (G + K)$  matrix, problematic in the current example with  $G + K = 4168$ . Thus, we illustrate the method below using only a subset of the households.

The large matrix inversion can be avoided using an analytical result, obtained through partitioned matrix inversion, that is applicable when the regressors include a set of dummy variables, such as the  $G$  dummies in (6.8). The `areg` command (and `xtreg, fe`) implements this simplification.

To contrast these various methods, we create a dataset with just two observations per cluster, here household. We first create a variable for the number of persons in each household.

```
. * Generate integer-valued person identifiers
. sort hh
. by hh: generate numpersonsinhh = _N
```

We then restrict analysis to the 441 households with exactly 2 members. The FE model is fit using 1) the `regress` command applied to the LSDV model, 2) the `areg` command, and 3) the `xtreg, fe` command.

For the `regress` command, the 441 dummy variables are entered as `i.hh`, leading to a large number of regressors and hence a large value for `matsize`. For the current example and using Stata 17, there is no problem. In versions of Stata before version 17, one may need to use the `set matsize` command to increase `matsize` from program defaults. In Stata 17, the `set matsize` command is redundant, but matrix sizes are limited by the edition of Stata; the `help limits` command gives these limits.

Both default and cluster-robust standard errors are obtained. We have

```

. * Various standard errors for hh FE estimation using xtreg, reg, and areg
. preserve
. keep if numpersonsinh == 2
(26,871 observations deleted)
. qui regress pharvis lnhhexp illness i.hh
. estimates store LSDV_def
. qui areg pharvis lnhhexp illness, absorb(hh)
. estimates store AREG_def
. qui xtreg pharvis lnhhexp illness, fe
. estimates store XTREG_def
. qui regress pharvis lnhhexp illness i.hh, vce(cluster hh)
. estimates store LSDV_clu
. qui areg pharvis lnhhexp illness, absorb(hh) vce(cluster hh)
. display "AREG se times square-root(2) = " _se[illness]/sqrt(2)
AREG se times square-root(2) = .12817079
. estimates store AREG_clu
. qui xtreg pharvis lnhhexp illness, fe vce(cluster hh)
. estimates store XTREG_clu
. estimates table LSDV_def AREG_def XTREG_def LSDV_clu AREG_clu XTREG_clu,
>      keep(illness lnhhexp _cons) b(%8.4f) se stats(N r2 df_m) varwidth(8)

```

Variable	LSDV_def	AREG_def	XTREG_def	LSDV_clu	AREG_clu	XTREG_clu
illness	0.8402 0.0949	0.8402 0.0949	0.8402 0.0949	0.8402 0.1813	0.8402 0.1813	0.8402 0.1282
lnhhexp	-0.3722 0.4035	(omitted) 0.0214	(omitted) 0.0214	-0.3722 0.0214	(omitted) 0.0214	(omitted) 0.0214
_cons	2.3636 1.3794	0.0705 0.1095	0.0705 0.1095	2.3636 0.1930	0.0705 0.1778	0.0705 0.1257
N	882	882	882	882	882	882
r2	0.6188	0.6188	0.1512	0.6188	0.6188	0.1512
df_m	441.0000	1.0000	441.0000	0.0000	1.0000	0.0000

Legend: b/se

```
. restore
```

The `areg` and `xtreg, fe` commands yield exactly the same coefficient estimates. The LSDV estimates fit using the `regress` command provide the same fit and the same coefficient for `illness`. The apparent differences in the other coefficients arise for the following reason. The LSDV intercept differs because the `areg` and `xtreg, fe` commands add in the grand means to estimate (6.7) rather than (6.6). The LSDV coefficient for `lnhhexp` is nonzero because, while `lnhhexp` is not identified, the `regress` command

does not know the source of nonidentification and instead dropped one of the household dummy variables.

The default standard errors for `illness` are the same across all three methods, and the default standard errors for the intercept are the same using the `areg` and `xtreg, fe` commands.

The important difference is in the cluster-robust standard errors. The `areg` command and LSDV estimation lead to much larger standard errors than the `xtreg, fe` command. In fact, in this example, the difference for `illness` is the multiple  $\sqrt{2} = 1.414$ .

The reason for this stark difference is the following. The cluster-robust estimate of the VCE uses the finite sample correction of  $\{G/(G - 1)\}\{(N - 1)/(N - K)\}$ , where  $K$  is the number of identified regressors and  $N = 882$  in this example. There are two identified regressors, the intercept and `illness`, so the `xtreg, fe` command sets  $K = 2$  and  $N - K = 882 - 2 = 880$ . The `areg` command, and `regress` used for the LSDV model, additionally counts the 440 identified dummy variables, leading to  $K^* = 442$  and  $N - K^* = 882 - 442 = 440$ . This is exactly one-half of  $N - K$  computed using the `xtreg, fe` command, leading to a VCE for LSDV and `areg` that is twice as large and standard errors that are  $\sqrt{2}$  times as large. For cluster-robust standard error correction, the dummy variables should not be counted in computing the degrees of freedom correction; one should use `xtreg, fe`.

More generally, the `areg` and LSDV standard errors are  $\sqrt{(N - K)/(N - G - 1 - K)}$  times the `xtreg, fe` standard errors, where  $K$  is the number of identified regressors in the within model ([\(6.6\)](#)).

### 6.6.5 Correlated RE model

The correlated RE model, explained more fully for the linear panel model in section [8.7.4](#), models the individual-specific effect as  $\alpha_g = \bar{x}'_{1,g}\gamma + \eta_g$ , where  $\eta_g$  is an i.i.d. error and  $\bar{x}_{1,g}$  are the within-cluster averages of those regressors that vary within cluster. Then the model [\(6.5\)](#) becomes

$$y_{gi} = \mathbf{x}'_{gi}\boldsymbol{\beta} + \bar{\mathbf{x}}_g'\boldsymbol{\gamma} + (\eta_g + \varepsilon_{gi})$$

This model is interpreted as an RE model in which the RE assumptions hold conditionally on both  $\mathbf{x}_{gi}$  and  $\bar{\mathbf{x}}_{1,g}$ . For the current example, we obtain

```

. * RE estimation with hh fixed
. sort hh
. by hh: egen avelnhhexp = mean(lnhhexp)
. by hh: egen aveillness = mean(illness)

. xtreg pharvis avelnhhexp aveillness lnhhexp illness, re vce(cluster hh)
note: lnhhexp omitted because of collinearity.

Random-effects GLS regression                               Number of obs      =     27,753
Group variable: hh                                         Number of groups   =      5,739
R-squared:                                                 Obs per group:
    Within = 0.1541                                         min =          1
    Between = 0.2071                                        avg =        4.8
    Overall = 0.1818                                       max =       19
                                                Wald chi2(3)     =    1370.89
corr(u_i, X) = 0 (assumed)                                Prob > chi2      =     0.0000
                                                               (Std. err. adjusted for 5,739 clusters in hh)

```

pharvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
avelnhhexp	.022084	.0144866	1.52	0.127	-.0063093	.0504773
aveillness	.0311645	.0284538	1.10	0.273	-.0246039	.0869329
lnhhexp	0	(omitted)				
illness	.6089755	.0179153	33.99	0.000	.5738621	.6440889
_cons	.0609077	.0393996	1.55	0.122	-.016314	.1381295
sigma_u	.62622406					
sigma_e	1.0606587					
rho	.25848181	(fraction of variance due to u_i)				

The coefficient of `illness` is the same as that for the FE estimator. The variable `lnhhexp` does not vary within cluster so is omitted; with a different ordering of regressors, `avelnhhexp` may be omitted instead.

This method can be useful in those nonlinear models for which there is no standard FE estimator in short panels because of the incidental parameters problem; see section 22.2.1.

A variant of this specification used in the peer group literature includes as regressor the average of  $x$  taken over a peer group or over a reference group analogous to a peer group. The intention is to control for the impact of the average characteristics of the peer or reference group, the latter being a subset of the sample under study. Manski's (1993) "linear-in-the-means" model includes as regressors averages of regressors of members of the peer group; this too can also be interpreted as controls for capturing cross-sectional dependence or social interdependence.

### 6.6.6 Methods to compute two-way FE models

Two-way FE models arise when fixed effects arise for two distinct reasons that are nonnested, such as for matched employer–employee data with potentially multiple observations for each employer and each employee.

If there are few fixed effects across both dimensions, one can use command `regress` with dummy variable regressors. Suppose the identifiers for the two dimensions are `id1` and `id2`. Then we can give command `regress y x i.id1 i.id2`.

If there are few fixed effects across one dimension, say, `id1`, and many across the other dimension, say, `id2`, the preceding method may not be practical because OLS regression then entails inverting a large matrix. This can be avoided by using the `areg` or `xtreg, fe` command, both of which implement the mean-differencing transformation (6.6). Then we can give command `areg y x i.id1, absorb(id2)` or the commands `xtset id2` and `xtreg y x i.id1, fe`. For cluster-robust inference, it is better to use `xtreg, fe` as explained in section 6.6.4.

If there are many fixed effects, then again estimation may no longer be practical because of OLS regression of the mean-differenced model requiring inversion of a large matrix. For example, if the smaller of the two dimensions has  $J$  distinct units, then we have at least a  $J \times J$  matrix to invert. Several methods have been proposed to avoid this matrix inversion, and there are several community-contributed Stata programs. These include `reg2hdf` ([Guimarães and Portugal 2010](#)) and `felsdvreg` ([Cornelissen 2008](#)). [McCaffrey et al. \(2012\)](#) provide a review.

## 6.7 Linear mixed models for clustered data

In OLS regression, there is no role for heterogeneity across individuals, aside from the error term  $u_i$ , which is a noise term assumed to be uncorrelated with regressors. The RE model (and the FE model) introduces some additional heterogeneity by allowing the intercept to vary across clusters.

The richer linear mixed model introduces still more heterogeneity by permitting slope parameters to vary across clusters. This can lead to more efficient estimation, and additionally, this approach is especially advantageous in settings where heterogeneity is of intrinsic interest. For example, we might be interested in the variability across villages in the response of pharmaceutical drugs to increased illness and not just the average response. Similarly, one may be interested in the variability across classrooms and schools of individual student performance to an educational intervention. And it is a natural way to more accurately reflect the richer data structure that arises in complex surveys.

Like the simpler RE model, the linear mixed model assumes that the random parameters are uncorrelated with the error term, so parameter estimates are inconsistent if the FE estimator is needed. If FE estimation is unnecessary, then linear mixed-model estimates of slope parameter means are consistent, and by allowing a richer specification of the random components of the model, the mixed linear model can lead to more efficient estimation. If the model for parameter heterogeneity is misspecified, however, then the linear mixed-model estimates of the variances and covariances of slope parameter estimates are inconsistent, and cluster-robust standard errors should be used rather than default standard errors.

The linear mixed model is widely used in other disciplines but is not often used by microeconometricians despite growing interest in applied economics research in heterogeneous responses. Reasons for this lack of use may include the microeconometrics focus on heterogeneity at the individual level, rather than at a cluster level, the desire to minimize distributional assumptions, and a focus on controlling for endogeneity with data that are usually observational.

### 6.7.1 Linear mixed model

The mixed linear model specifies a model for the conditional variance and covariances of  $y_{gi}$  that can depend on observable variables.

For individual  $i$  in cluster  $g$  the conditional mean of  $y_{gi}$  is specified to be  $\mathbf{x}'_{gi}\beta$ , where the regressors  $\mathbf{x}_{gi}$  include an intercept.

The simplest version of the mixed linear model is a two-level model, where the two levels are the individual and the cluster. Then the observed value  $y_{gi}$  equals this conditional mean plus an error term  $\mathbf{z}'_{gi}\mathbf{u}_g + \varepsilon_{gi}$ , where  $\mathbf{z}_{gi}$  are observable variables and  $\mathbf{u}_g$  and  $\varepsilon_{gi}$  are i.i.d. normally distributed random variables. We have

$$y_{gi} = \mathbf{x}'_{gi}\beta + \mathbf{z}'_{gi}\mathbf{u}_g + \varepsilon_{gi} \quad (6.9)$$

where  $\mathbf{u}_g \sim N(\mathbf{0}, \Sigma_u)$  and  $\varepsilon_{gi} \sim N(0, \sigma_\varepsilon^2)$ . The variances and covariances in  $\Sigma_u$  are called RE parameters.

The mixed models literature refers to the conditional mean parameters  $\beta$  as fixed effects, which are contrasted with the error terms  $\mathbf{u}_g$ , which are called random effects. We minimize use of this terminology because this is a very different use of the term “fixed effects” from that typically used by economists. Indeed, if the fixed effects defined in section 6.6 are present, then the estimators of this section are inconsistent.

Specific choices of  $\mathbf{z}_{gi}$  lead to some standard models. OLS corresponds to  $\mathbf{z}_{gi} = 0$ . The RE model of section 6.5 corresponds to  $\mathbf{z}_{gi} = 1$  because only the intercept is random. A model often called the random-coefficients model sets  $\mathbf{z}_{gi} = \mathbf{x}_{gi}$  so that, for the regressor  $\mathbf{x}_{gi}$ , both the intercept and slope coefficients are random. The hierarchical linear models framework (see section 6.7.7) leads to further choices of  $\mathbf{z}_{gi}$ .

### 6.7.2 Linear mixed-model estimator

Mixed linear models can have several levels. Regardless of the number of levels, the general form of the model upon stacking all  $N$  observations across all levels is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}$$

where  $\mathbf{u}$  and  $\boldsymbol{\varepsilon}$  are zero-mean multivariate normal errors that are independent of each other.

The combined error term  $\mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}$  is therefore multivariate normal, with variance matrix that, for the two-level model (6.9), is parameterized by  $\Sigma_{\mathbf{u}}$  and  $\sigma_{\varepsilon}^2$ . This is a natural setting for FGLS estimation, or one can use the MLE given the assumption of normality.

The `mixed` command estimates by ML, the default, or by an asymptotically equivalent variation of ML, called restricted maximum likelihood, that produces variance estimates that are unbiased in balanced samples. The `dfmethod()` option specifies different methods for computing the degrees of freedom.

While homoskedasticity and normality of the errors  $\mathbf{u}_g$  and  $\boldsymbol{\varepsilon}_{gi}$  are assumed, it is not necessary for consistency of  $\boldsymbol{\beta}$ , because consistency requires essentially that  $E(\mathbf{y}|\mathbf{X}, \mathbf{Z}) = \mathbf{X}\boldsymbol{\beta}$ , which in turn requires that  $\mathbf{u}_g$  and  $\boldsymbol{\varepsilon}_{gi}$  have mean zero.

Despite this robustness of the mixed estimator of  $\boldsymbol{\beta}$ , homoskedasticity of the errors  $\mathbf{u}_g$  and  $\boldsymbol{\varepsilon}_{gi}$  is needed to obtain consistent estimates of the variances and covariances measuring heterogeneity. It is also needed for default standard errors to be correct. Otherwise, one should obtain cluster-robust standard errors that are valid even if the errors are not i.i.d. These robust standard errors are valid provided the errors are independent across clusters and the number of clusters is large.

### 6.7.3 The `mixed` command

The `mixed` command fits a multilevel mixed-effects model. The dependent variable and regressors in (6.9) are defined, followed by two vertical bars (`||`) and a definition of the portion of the model for the random effects  $\mathbf{u}_g$ .

For example, in a single-level model, the command `mixed y x || id: z, mle` is used if we want to regress  $y_{gi}$  on an intercept and  $x_{gi}$ ; the variable `id` identifies cluster  $g$ ; the random effects vary with an intercept and variable  $z_{gi}$ ; and estimation is by ML.

The general syntax of the command is

```
mixed depvar fe_equation [|| re_equation] [|| re_equation ...]
    [, options]
```

This general form allows for additional levels of clustering; see section [6.7.7](#).

The dependent variable  $y_{gi}$  is given in `depvar`. The regressors are defined in `fe_equation`, which has the syntax

```
[indepvars] [if] [in] [, fe_options]
```

where `indepvars` defines the regressors  $\mathbf{x}_{gi}$ , and the `fe_option` `noconstant` is added if no intercept is to be included.

The RE model is given in `re_equation`, which for random coefficients and intercepts has the syntax

```
levelvar: [varlist] [, re_options]
```

where `levelvar` is the individual unit identifier, `varlist` gives the variable  $\mathbf{z}_{gi}$ , and the `noconstant` `re_option` is added if there is to be no random intercept. For random effects among the values of a factor variable, `re_equation` has the alternative syntax

```
levelvar: R.varname [, re_options]
```

The `re_option` `covariance(vartype)` places structure on  $\Sigma_u$ , where `vartype` includes `independent` (the default) with  $\Sigma_u$  diagonal and `unstructured` with no structure placed on  $\Sigma_u$ .

## 6.7.4 Random intercept model

The random-intercept model restricts  $\mathbf{z}'_{gi} \mathbf{u}_g = u_g$ , so  $u_g$  is a scalar and  $z_{gi}$  is a scalar equal to one, and is identical to the RE model of section [6.5.3](#).

The model can be fit by using the `mixed` command. For a random intercept varying by `hh`, the RE portion of the model is defined simply as `hh::`. Here `hh` identifies the cluster, and no variables are listed after the colon because the default is to include a random intercept. We use the `mle` and `vce(robust)` options to obtain

```
. * Random hh intercept model estimated using mixed
. mixed pharvis lnhhexp illness || hh: , mle vce(robust)

Performing EM optimization ...

Performing gradient-based optimization:
Iteration 0:  log pseudolikelihood = -43424.009
Iteration 1:  log pseudolikelihood = -43423.989
Iteration 2:  log pseudolikelihood = -43423.989

Computing standard errors ...

Mixed-effects regression                               Number of obs      =     27,753
Group variable: hh                                  Number of groups   =      5,739
                                                       Obs per group:
                                                       min =           1
                                                       avg =         4.8
                                                       max =        19
                                                       Wald chi2(2)    =     1358.18
Log pseudolikelihood = -43423.989                  Prob > chi2       =     0.0000
                                                       (Std. err. adjusted for 5,739 clusters in hh)
```

pharvis	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
lnhhexp	.0192194	.0147418	1.30	0.192	-.009674 .0481128
illness	.6178673	.0167863	36.81	0.000	.5849668 .6507679
_cons	.0815646	.0393568	2.07	0.038	.0044266 .1587026

Random-effects parameters	Estimate	Robust	std. err.	[95% conf. interval]
hh: Identity				
var(_cons)	.2743023	.0337261	.2155621	.3490491
var(Residual)	1.146892	.0849008	.9919978	1.325972

The coefficient estimates are similar (within 5%) to those from the `xtreg, re` command in section [6.5](#). And, as shown below, they are identical to those obtained from the `xtreg, mle` command. The cluster-specific error  $u_j$  has estimated variance 0.274, and the individual-specific error  $\varepsilon_{gi}$  has variance 1.147.

For this example, the `reml` option of `mixed` yields coefficient estimates that are the same to the fourth significant digit. The `reml` option does not support robust standard errors.

The default is for statistical inference to be based on standard normal  $p$ -values and critical values. In the special case where default standard errors are used, rather than robust standard errors, the `dfmethod()` option uses the  $t$  distribution with several different methods available to compute the degrees of freedom.

### 6.7.5 Cluster–robust standard errors for mixed estimator

We compare cluster–robust standard errors following the `mixed` command at broadening levels of clustering. Additionally, we demonstrate equivalence with the `xtreg, mle` command. We have

```
. * Robust standard errors after mixed (and equivalence with xtreg, mle)
. qui xtreg pharvis lnhhexp illness, mle
. estimates store remle_def
. qui mixed pharvis lnhhexp illness || hh: , mle
. estimates store mixed_def
. qui mixed pharvis lnhhexp illness || hh: , mle vce(robust)
. estimates store mixed_rob
. qui mixed pharvis lnhhexp illness || hh: , mle vce(cluster hh)
. estimates store mixed_hh
. qui mixed pharvis lnhhexp illness || hh: , mle vce(cluster commune)
. estimates store mixed_comm
```

```
. estimates table remle_def mixed_def mixed_rob mixed_hh mixed_comm,
>      b(%10.4f) se stats(N)
```

Variable	remle_def	mixed_def	mixed_rob	mixed_hh	mixed_comm
pharvis					
lnhhexp	0.0192 0.0154	0.0192 0.0154	0.0192 0.0147	0.0192 0.0147	0.0192 0.0215
illness	0.6179 0.0082	0.6179 0.0082	0.6179 0.0168	0.6179 0.0168	0.6179 0.0303
_cons	0.0816 0.0413	0.0816 0.0413	0.0816 0.0394	0.0816 0.0394	0.0816 0.0575
sigma_u					
_cons	0.5237 0.0101				
sigma_e					
_cons	1.0709 0.0051				
lns1_1_1					
_cons		-0.6468 0.0193	-0.6468 0.0615	-0.6468 0.0615	-0.6468 0.0776
lnsig_e					
_cons		0.0685 0.0048	0.0685 0.0370	0.0685 0.0370	0.0685 0.0486
Statistics					
N	27753	27753	27753	27753	27753

Legend: b/se

The first two columns demonstrate that the `xtreg, mle` command yields the same estimates and standard errors for the estimate of  $\beta$  as the `mixed` command. The error variances are also identical because, for example,  $\ln(0.5237) = -0.6468$ .

The standard errors increase as the level of clustering broadens, and because the clustering was on household, the `vce(robust)` option yields the same standard errors as the `vce(cluster hh)` option.

## 6.7.6 Random slopes model

Although the regression parameters  $\beta$  in (6.9) are consistently estimated if  $\mathbf{u}$  and  $\varepsilon_{gi}$  are not i.i.d., the estimates of the variance parameters  $\Sigma_u$  and  $\sigma_\varepsilon$  (reported here as `var(_cons)` and `var(Residual)`) are inconsistently estimated. And there is efficiency loss in estimation of  $\beta$ .

This provides motivation for using a richer model for the RE portion of the model.

For our application, we let the random effects depend on both an intercept and on `illness`, and we let  $\Sigma_u$  be unstructured. Using default standard errors, we obtain

```
. * Random-slopes model estimated using mixed  
. mixed pharvis lnhhexp illness || hh: illness, mle covar(unstructured)  
>      difficult noheader nolog
```

pharvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lnhhexp	.0035616	.0098921	0.36	0.719	-.0158265 .0229497
illness	.7687124	.015792	48.68	0.000	.7377607 .7996641
_cons	.0649558	.0269171	2.41	0.016	.0121993 .1177123

Random-effects parameters	Estimate	Std. err.	[95% conf. interval]	
hh: Unstructured				
var(illness)	.8119174	.0236232	.7669121	.8595637
var(_cons)	.0001208	.00005	.0000536	.0002721
cov(illness, _cons)	.009903	.0020559	.0058735	.0139324
var(Residual)	.7092376	.0066412	.6963399	.7223742

LR test vs. linear model: chi2(3) = 10653.94 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

The first set of output gives the estimated coefficients of the intercept and slope parameters. From the second set of output, all the variance components parameters are statistically significantly different from 0 at 5%. Furthermore, the heterogeneity is very large. For example, the slope coefficient for illness has standard deviation across households of  $\sqrt{0.812} = 0.901$ , which is large compared with the mean of 0.769.

Because default standard errors were used, the output includes a joint test that the three RE parameters are zero. The joint-test statistic has a nonstandard and complicated distribution because a variance must be positive, so the  $\chi^2(3)$  distribution is an approximation that overstates the  $p$ -values. Because this conservative value of  $p$  is less than 0.05 in this example, we can definitely reject the null hypothesis at level 0.05. The likelihood-ratio joint test statistic is not computed if the `vce(robust)` option is used. Instead, a Wald test needs to be implemented; see section 23.4 for an example.

Numerical problems can be encountered when richer models for the random effects are specified. For example, dropping the `covar(unstructured)` option leads to `var(_cons)` going to its boundary value of zero.

Interest can lie in predicting the random effects for each group and in predicting the conditional mean when conditioning on both regressors. For an individual observation,  $y_{gi} = \mathbf{x}'_{gi}\beta + \mathbf{z}'_{gi}\mathbf{u}_g + \varepsilon_{gi}$ . The `reffects` option of the `predict` postestimation command computes best linear unbiased predictions  $\hat{\mathbf{u}}_g$ , and the `fitted` option computes  $\hat{y}_{gi} = \mathbf{x}'_{gi}\hat{\beta} + \mathbf{z}'_{gi}\hat{\mathbf{u}}_g$ . The default `xb` option of `predict` instead computes  $\hat{y}_{gi} = \mathbf{x}'_{gi}\hat{\beta}$ .

In the linear mixed model,  $E(y_{gi}|\mathbf{x}_{gi}) = \mathbf{x}'_{gi}\beta$ , so the `margins` command computes marginal means  $\mathbf{x}'_{gi}\hat{\beta}$  and marginal effects  $\beta$ .

### 6.7.7 Hierarchical linear models

The mixed model presented so far has been a two-level model. Mixed models or hierarchical models allow additional levels.

A three-level example is to suppose that person  $i$  is in household  $g$ , which is in village  $k$ , and that the model is a variance-components model with

$$y_{gki} = \mathbf{x}'_{gki}\beta + u_g + v_k + \varepsilon_{gki}$$

where  $u_g$ ,  $v_k$ , and  $\varepsilon_{gki}$  are i.i.d. errors.

The model can be fit using the `mixed` command, with the variance components ordered beginning with the broadest level, here `commune` because households are nested in villages. The `difficult` option was added to ensure convergence of the iterative process. We obtain

. * Hierarchical linear model with household and commune variance components						
. mixed pharvis lnhhexp illness    commune:    hh:, mle difficult						
> nolog vce(cluster commune)						
Mixed-effects regression				Number of obs	=	27,753
Grouping information						
	Group variable	No. of groups	Observations per group			
			Minimum	Average	Maximum	
	commune	194	51	143.1	206	
	hh	5,739	1	4.8	19	
				Wald chi2(2)	=	406.45
Log pseudolikelihood = -43200.141				Prob > chi2	=	0.0000
			(Std. err. adjusted for 194 clusters in commune)			
	pharvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
	lnhhexp	-.0409411	.0206685	-1.98	0.048	-.0814506 -.0004317
	illness	.6140699	.0305092	20.13	0.000	.5542729 .6738668
	_cons	.2353656	.0568166	4.14	0.000	.1240072 .346724
	Random-effects parameters		Robust Estimate	std. err.	[95% conf. interval]	
commune: Identity						
	var(_cons)		.065903	.0107784	.0478289	.090807
hh: Identity						
	var(_cons)		.2042623	.0335477	.1480431	.2818307
	var(Residual)		1.148946	.1117647	.9495068	1.390275

The coefficient of `lnhhexp` has changed sign compared with the RE estimates. Both variance components are statistically significant at level 0.05, with the household-specific error having much larger variance than the commune-specific error.

The `mixed` command allows the variance components to additionally depend on regressors, as already demonstrated in the simpler case of a two-level model.

### 6.7.8 Two-way RE model

The `mixed` command is intended for multilevel models where the levels are nested. We now consider a two-way RE model that has the error  $\alpha_g + \gamma_k + \varepsilon_{gki}$ , where all three errors are i.i.d. and now  $g$  is not nested in  $k$  and  $k$  is not nested in  $g$ .

[Rabe-Hesketh and Skrondal \(2022\)](#), 488) explain how to nonetheless use `mixed` to estimate the parameters of the two-way RE model, using a result of [Goldstein \(1987\)](#) that shows how to rewrite covariance models with nonnested structure as nested models.

Let the variables defining  $g$  and  $k$  be `lev1` and `lev2`. Then, we specify the two levels as `|| _all: R.lev2 || lev1::`. The entry `_all:` defines each combination  $(g, k)$  to be a separate group (thereby nesting `lev1`). We then add `R.lev2` because the `R.` prefix ensures the desired correlation pattern due to  $\gamma_k$  by defining a factor structure in  $g$  with independent factors with identical variance (see [ME] **mixed**). At the second level, the RE equation defines the covariance structure for  $\alpha_g$  by simply using `lev1::`. This ordering is optimal when there are more different values of `lev1` than `lev2`. If this is not the case, then computation is faster if the roles of  $g$  and  $k$  are reversed, and the random effects are specified as `|| _all: R.lev1 || lev2::`.

For purely illustrative purposes, we cluster in two nonnested dimensions: `commune` and `illness`. There are 194 communes and only 10 distinct values of `illness`, so it is fastest to use the ordering `|| _all: R.illness || commune::`. We obtain

```
. * Two-way nonnested errors - illness and commune
. mixed pharvis lnhhexp illness || _all: R.illness || commune: ,
>     mle difficult nolog
```

Mixed-effects ML regression Number of obs = 27,753

Grouping information

Group variable	No. of groups	Observations per group		
		Minimum	Average	Maximum
_all	1	27,753	27,753.0	27,753
commune	194	51	143.1	206

Log likelihood = -43194.504	Wald chi2(2) = 14.91
	Prob > chi2 = 0.0006

pharvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lnhhexp	-.0383132	.0146749	-2.61	0.009	-.0670754 -.0095509
illness	.3714657	.1306934	2.84	0.004	.1153114 .62762
_cons	.6637303	.5692554	1.17	0.244	-.4519898 1.77945

Random-effects parameters	Estimate	Std. err.	[95% conf. interval]
_all: Identity var(R.illness)	.8275655	.5503089	.2247895 3.046693
commune: Identity var(_cons)	.0656064	.007628	.0522369 .0823976
var(Residual)	1.295374	.0110375	1.273921 1.317189

LR test vs. linear model: chi2(2) = 1902.37 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

The coefficient of `lnhhexp` has changed sign compared with the RE estimates. Both variance components are statistically significant at level 0.05, with the illness-specific error having much larger variance than the commune-specific error.

The same results were obtained with the ordering `|| _all: R.commune || illness:`, but computation took much longer.

In the two-way nonnested case, the variance matrix of the combined errors  $\alpha_g + \gamma_k + \varepsilon_{gki}$  is not block diagonal, so it is not possible to compute robust standard errors.

## 6.8 Systems of linear regressions

In this section, we extend GLS estimation to a system of linear equations with errors that are correlated across equations for a given individual but are uncorrelated across individuals. Then cross-equation correlation of the errors can be exploited to improve estimator efficiency. This multivariate linear regression model is usually referred to in econometrics as a set of SUR equations. It arises naturally in many contexts in economics—a system of demand equations is a leading example. The GLS methods presented here can be extended to systems of simultaneous equations (three-stage least-squares estimation presented in section 7.10), panel data (chapter 8), and systems of nonlinear equations (section 18.11.2).

We also illustrate how to test or impose restrictions on parameters across equations. This additional complication can arise with systems of equations. For example, consumer demand theory may impose symmetry restrictions.

### 6.8.1 Seemingly unrelated regressions model

The model consists of  $m$  linear regression equations for  $N$  individuals. The  $j$ th equation for individual  $i$  is  $y_{ji} = \mathbf{x}'_{ji}\boldsymbol{\beta}_j + u_{ji}$ . With all observations stacked, the model for the  $j$ th equation can be written as  $\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta}_j + \mathbf{u}_j$ . We then stack the  $m$  equations to give the SUR model

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_m \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_2 & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{X}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_m \end{bmatrix} + \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_m \end{bmatrix}$$

This has a compact representation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

The error terms are assumed to have zero mean and to be independent across individuals and homoskedastic. The complication is that for a given individual, the errors are correlated across equations, with

$E(u_{ij}u_{ij'}|\mathbf{X}) = \sigma_{jj'}$  and  $\sigma_{jj'} \neq 0$  when  $j \neq j'$ . It follows that the  $N \times 1$  error vectors  $\mathbf{u}_j$ ,  $j = 1, \dots, m$ , satisfy the assumptions 1)  $E(\mathbf{u}_j|\mathbf{X}) = \mathbf{0}$ ; 2)  $E(\mathbf{u}_j\mathbf{u}_j'|\mathbf{X}) = \sigma_{jj}\mathbf{I}_N$ ; and 3)  $E(\mathbf{u}_j\mathbf{u}_{j'}'|\mathbf{X}) = \sigma_{jj'}\mathbf{I}_N$ ,  $j \neq j'$ . Then for the entire system,  $\Omega = E(\mathbf{u}\mathbf{u}') = \Sigma \otimes \mathbf{I}_N$ , where  $\Sigma$  is an  $m \times m$  positive-definite matrix with  $jj'$ 'th entry  $\sigma_{jj'}$  and  $\otimes$  denotes the Kronecker product of two matrices.

OLS applied to each equation yields a consistent estimator of  $\beta$ , but the optimal estimator for this model is the GLS estimator. Using

$\Omega^{-1} = \Sigma^{-1} \otimes \mathbf{I}_N$ , because  $\Omega = \Sigma \otimes \mathbf{I}_N$ , we see the GLS estimator is

$$\hat{\beta}_{\text{GLS}} = \{\mathbf{X}' (\Sigma^{-1} \otimes \mathbf{I}_N) \mathbf{X}\}^{-1} \{\mathbf{X}' (\Sigma^{-1} \otimes \mathbf{I}_N) \mathbf{y}\} \quad (6.10)$$

with a vce assuming homoskedastic errors independent over  $i$  that is given by

$$\text{Var}(\hat{\beta}) = \{\mathbf{X}' (\Sigma^{-1} \otimes \mathbf{I}_N) \mathbf{X}\}^{-1}$$

FGLS estimation is straightforward, and the estimator is called the SUR estimator. We require only estimation and inversion of the  $m \times m$  matrix  $\Sigma$ . Computation is in two steps. First, each equation is estimated by OLS, and the residuals from the  $m$  equations are used to estimate  $\Sigma$ , using  $\hat{\mathbf{u}}_j = \mathbf{y}_j - \mathbf{X}_j \hat{\beta}_j$  and  $\hat{\sigma}_{jj'} = \hat{\mathbf{u}}_j' \hat{\mathbf{u}}_{j'}/N$ . Second,  $\hat{\Sigma}$  is substituted for  $\Sigma$  in (6.10) to obtain the FGLS estimator  $\hat{\beta}_{\text{FGLS}}$ . An alternative is to further iterate these two steps until the estimation converges, called the iterated FGLS estimator. Although asymptotically there is no advantage from iterating, in finite samples there may be. Asymptotic theory assumes that  $m$  is fixed while  $N \rightarrow \infty$ .

There are two cases where FGLS reduces to equation-by-equation OLS. The first is the obvious case of errors uncorrelated across equations, so  $\Sigma$  is diagonal. The second case is less obvious but can often arise in practice. Even if  $\Sigma$  is nondiagonal, if each equation contains exactly the same set of regressors, so  $\mathbf{X}_j = \mathbf{X}_{j'}$  for all  $j$  and  $j'$ , then it can be shown that the FGLS systems estimator reduces to equation-by-equation OLS.

### 6.8.2 The `sureg` command

The SUR estimator is performed in Stata by using the command `sureg`. This command requires specification of dependent and regressor variables for each of the  $m$  equations. The basic syntax for `sureg` is

```
sureg (depvar1 varlist1) (depvar2 varlist2) ... (depvarN varlistN) [if] [in] [weight]
```

where each pair of parentheses contains the model specification for each of the  $m$  linear regressions. The default is two-step SUR estimation. Specifying the `isure` option causes `sureg` to produce the iterated estimator. For robust VCE, see section [6.8.4](#).

### 6.8.3 Application to two categories of expenditures

The application of SUR considered here involves two dependent variables that are the logarithm of expenditure on prescribed drugs (`1drugexp`) and expenditure on all categories of medical services other than drugs (`1totothr`).

This data extract from the Medical Expenditure Panel Survey is similar to that studied in chapter [3](#) and covers the Medicare-eligible population of those aged 65 years and more. The regressors are socioeconomic variables (`educyr` and a quadratic in `age`), health-status variables (`actlim` and `totchr`), and supplemental insurance indicators (`private` and `medicaid`). We have

```

. * Summary statistics for seemingly unrelated regressions example
. clear all
. qui use mus206mepssur
. summarize ldrugexp ltotothr age age2 educyr actlim totchr medicaid private

```

Variable	Obs	Mean	Std. dev.	Min	Max
ldrugexp	3,285	6.936533	1.300312	1.386294	10.33773
ltotothr	3,350	7.537196	1.61298	1.098612	11.71892
age	3,384	74.38475	6.388984	65	90
age2	3,384	5573.898	961.357	4225	8100
educyr	3,384	11.29108	3.7758	0	17
actlim	3,384	.3454492	.4755848	0	1
totchr	3,384	1.954492	1.326529	0	8
medicaid	3,384	.161643	.3681774	0	1
private	3,384	.5156619	.4998285	0	1

The parameters of the SUR model are estimated by using the `sureg` command. Because SUR estimation reduces to OLS if exactly the same set of regressors appears in each equation, we omit `educyr` from the model for `ldrugexp`, and we omit `medicaid` from the model for `ltotothr`. We use the `corr` option because this yields the correlation matrix for the fitted residuals that is used to form a test of the independence of the errors in the two equations. We have

```

. * SUR estimation of a seemingly unrelated regressions model
. sureg (ldrugexp age age2 actlim totchr medicaid private)
>      (ltotothr age age2 educyr actlim totchr private), corr
Seemingly unrelated regression

```

Equation	Obs	Params	RMSE	"R-squared"	chi2	P>chi2
ldrugexp	3,251	6	1.133657	0.2284	962.07	0.0000
ltotothr	3,251	6	1.491159	0.1491	567.91	0.0000

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
ldrugexp	age	.2630418	.0795316	3.31	0.001
	age2	-.0017428	.0005287	-3.30	0.001
	actlim	.3546589	.046617	7.61	0.000
	totchr	.4005159	.0161432	24.81	0.000
	medicaid	.1067772	.0592275	1.80	0.071
	private	.0810116	.0435596	1.86	0.063
	_cons	-3.891259	2.975898	-1.31	0.191
ltotothr	age	.2927827	.1046145	2.80	0.005
	age2	-.0019247	.0006955	-2.77	0.006
	educyr	.0652702	.00732	8.92	0.000
	actlim	.7386912	.0608764	12.13	0.000
	totchr	.2873668	.0211713	13.57	0.000
	private	.2689068	.055683	4.83	0.000
	_cons	-5.198327	3.914053	-1.33	0.184

Correlation matrix of residuals:

ldrugexp	ltotothr
ldrugexp	1.0000
ltotothr	0.1741
	1.0000

Breusch-Pagan test of independence: chi2(1) = 98.590, Pr = 0.0000

There are only 3,251 observations in this regression because of missing values for ldrugexp and ltotothr. The lengthy output from sureg has three components.

The first set of results summarizes the goodness-of-fit for each equation. For the dependent variable ldrugexp, we have  $R^2 = 0.23$ . A test for joint significance of all regressors in the equation (aside from the intercept) has a value of 962.07 with a  $p$ -value of  $p = 0.000$  obtained from the  $\chi^2(6)$

distribution because there are 6 regressors. The regressors are jointly significant in each equation.

The middle set of results presents the estimated coefficients. Most regressors are statistically significant at the 5% level, and the regressors generally have a bigger impact on other expenditures than they do on drug expenditures. As you will see in exercise 7 at the end of this chapter, the coefficient estimates are similar to those from OLS, and the efficiency gains of SUR compared with OLS are relatively modest, with standard errors reduced by roughly 1%.

The final set of results are generated by the `corr` option. The errors in the two equations are positively correlated, with  $r_{12} = \hat{\sigma}_{12}/\sqrt{\hat{\sigma}_{11}\hat{\sigma}_{22}} = 0.1741$ . The Breusch–Pagan Lagrange multiplier test for error independence, computed as  $Nr_{12}^2 = 3251 \times 0.1741^2 = 98.54$ , has  $p = 0.000$  using the  $\chi^2(1)$  distribution. (Because  $r_{12}$  is not exactly equal to 0.1741, the hand calculation yields 98.54, which does not exactly equal 98.590 in the output.) There is statistically significant correlation between the errors in the two equations, as should be expected because the two categories of expenditures may have similar underlying determinants. At the same time, the correlation is not particularly strong, so the efficiency gains to SUR estimation are not great in this example.

#### 6.8.4 Robust standard errors for the SUR estimator

The standard errors reported from `sureg` impose homoskedasticity. This is a reasonable assumption in this example because taking the natural logarithm of expenditures greatly reduces heteroskedasticity. But in other applications, such as using the levels of expenditures, this would not be reasonable.

There is no option available with `sureg` to allow the errors to be heteroskedastic. However, the `bootstrap` and `jackknife` prefixes, explained in chapter 12, can be used. The `bootstrap` prefix resamples over individuals and provides standard errors that are valid under the weaker assumption that  $E(u_{ji}u_{j'i}|\mathbf{X}) = \sigma_{jj',i}$ , while maintaining the assumption of independence over individuals. As explained in section 12.3.4, it is good practice to use more bootstraps than the Stata default and to set a seed. We have

```

. * Bootstrap to get heteroskedasticity-robust standard errors for SUR estimator
. bootstrap, reps(400) seed(10101) nodots: sureg
>      (ldrugexp age age2 actlim totchr medicaid private)
>      (ltotothr age age2 educyr actlim totchr private)

```

Seemingly unrelated regression

Equation	Obs	Params	RMSE	"R-squared"	chi2	P>chi2
ldrugexp	3,251	6	1.133657	0.2284	962.07	0.0000
ltotothr	3,251	6	1.491159	0.1491	567.91	0.0000

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
ldrugexp	.2630418	.0743481	3.54	0.000	.1173222	.4087614
	-.0017428	.0004929	-3.54	0.000	-.0027089	-.0007766
	.3546589	.0462869	7.66	0.000	.2639382	.4453795
	.4005159	.0169809	23.59	0.000	.3672339	.4337979
	.1067772	.0642814	1.66	0.097	-.019212	.2327664
	.0810116	.044791	1.81	0.071	-.0067771	.1688002
	-3.891259	2.794579	-1.39	0.164	-9.368532	1.586015
ltotothr	.2927827	.1062298	2.76	0.006	.0845762	.5009892
	-.0019247	.0007048	-2.73	0.006	-.0033061	-.0005434
	.0652702	.0075052	8.70	0.000	.0505602	.0799802
	.7386912	.0619353	11.93	0.000	.6173003	.8600821
	.2873668	.0202824	14.17	0.000	.247614	.3271196
	.2689068	.0548669	4.90	0.000	.1613696	.376444
	-5.198327	3.991338	-1.30	0.193	-13.02121	2.624553

The output shows that the bootstrap standard errors differ little from the default standard errors. So, as expected for this example for expenditures in logs, heteroskedasticity makes little difference to the standard errors.

If data are clustered, with many clusters, then cluster-robust standard errors can be obtained by using a pairs cluster bootstrap.

### 6.8.5 Testing cross-equation constraints

Testing and imposing cross-equation constraints is possible using SUR estimation. We begin with testing.

To test the joint significance of the age regressors, we type

```

. * Test of variables in both equations
. qui sureg (ldrugexp age age2 actlim totchr medicaid private)
>          (ltotothr age age2 educyr actlim totchr private)
. test age age2
( 1) [ldrugexp]age = 0
( 2) [ltotothr]age = 0
( 3) [ldrugexp]age2 = 0
( 4) [ltotothr]age2 = 0
      chi2( 4) =    16.55
      Prob > chi2 =    0.0024

```

This command automatically conducted the test for both equations.

The format used to refer to coefficient estimates when more than one equation is estimated is `[depname] varname`, where *depname* is the name of the dependent variable in the equation of interest and *varname* is the name of the regressor of interest. For example, the preceding `sureg` output provides the name `ldrugexp` in the first equation output. Alternatively, the command `sureg, coeflegend` provides complete names for each coefficient, such as `[ldrugexp]age`.

A test for significance of regressors in just the first equation is therefore

```

. * Test of variables in just the first equation
. test [ldrugexp]age [ldrugexp]age2
( 1) [ldrugexp]age = 0
( 2) [ldrugexp]age2 = 0
      chi2( 2) =    10.98
      Prob > chi2 =    0.0041

```

The quadratic in `age` in the first equation is jointly statistically significant at the 5% level.

Now consider a test of a cross-equation restriction. Suppose we want to test the null hypothesis that having private insurance has the same impact on both dependent variables. We can set up the test as follows:

```

. * Test of a restriction across the two equations
. test [ldrugexp]private = [ltotothr]private
( 1) [ldrugexp]private - [ltotothr]private = 0
      chi2( 1) =     8.35
      Prob > chi2 =    0.0038

```

The null hypothesis is rejected at the 5% significance level. The coefficients in the two equations differ.

### 6.8.6 Imposing cross-equation constraints

We now obtain estimates that impose restrictions on parameters across equations. Usually, such constraints are based on economic theory. As an illustrative example, we impose the constraint that having private insurance has the same impact on both dependent variables.

We first use the `constraint` command to define the constraint.

- . \* Specify a restriction across the two equations
- . constraint 1 [ldrugexp]private = [ltotothr]private

Subsequent commands imposing the constraint will refer to it by the number 1 (any integer between 1 and 1,999 can be used).

We then impose the constraint using the `constraints()` option. We have

```

. * Estimate subject to the cross-equation constraint
. sureg (ldrugexp age age2 actlim totchr medicaid private)
>      (ltotothr age age2 educyr actlim totchr private), constraints(1)
Seemingly unrelated regression

```

Equation	Obs	Params	RMSE	"R-squared"	chi2	P>chi2
ldrugexp	3,251	6	1.134035	0.2279	974.09	0.0000
ltotothr	3,251	6	1.492163	0.1479	559.71	0.0000

( 1) [ldrugexp]private - [ltotothr]private = 0

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
ldrugexp	.2707053	.0795434	3.40	0.001	.1148031 .4266076
	-.0017907	.0005288	-3.39	0.001	-.0028271 -.0007543
	.3575386	.0466396	7.67	0.000	.2661268 .4489505
	.3997819	.0161527	24.75	0.000	.3681233 .4314405
	.1473961	.0575962	2.56	0.010	.0345096 .2602827
	.1482936	.0368364	4.03	0.000	.0760955 .2204917
	-4.235088	2.975613	-1.42	0.155	-10.06718 1.597006
ltotothr	.2780287	.1045298	2.66	0.008	.073154 .4829034
	-.0018298	.0006949	-2.63	0.008	-.0031919 -.0004677
	.0703523	.0071112	9.89	0.000	.0564147 .0842899
	.7276336	.0607791	11.97	0.000	.6085088 .8467584
	.2874639	.0211794	13.57	0.000	.245953 .3289747
	.1482936	.0368364	4.03	0.000	.0760955 .2204917
	-4.62162	3.910453	-1.18	0.237	-12.28597 3.042727

As desired, the `private` variable has the same coefficient (0.148) in the two equations.

More generally, separate `constraint` commands can be typed to specify many constraints, and the `constraints()` option will then have as an argument a list of the constraint numbers.

### 6.8.7 The `suest` command for seemingly unrelated equations

The SUR estimator jointly estimates equations by FGLS. Joint estimation has the benefit of potential efficiency gains and of allowing test of cross-equation restrictions.

Often, however, we wish to test cross-equation restrictions in equations that are separately estimated. This can be done using the postestimation command `suest`, an acronym for seemingly unrelated estimations. The basic syntax for `suest` is

```
suest namelist [ , options ]
```

where `namelist` is a list of names of model results previously stored using `estimates store` and options include `vce(robust)` and `vce(cluster clustvar)`. The model results need not necessarily be from the same estimation command.

As an example, consider separate OLS estimation of the models for `ldrugexp` and `ltotothr`, and then test that the coefficient for `private` is the same in the two equations. We have

```

. * suest used for cross-equations test of separately estimated models
. qui regress ldrugexp age age2 actlim totchr medicaid private
. estimates store DRUG
. qui regress ltotothr age age2 educyr actlim totchr private
. estimates store OTHER
. suest DRUG OTHER

```

Simultaneous results for DRUG, OTHER Number of obs = 3,384

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
DRUG_mean					
age	.2764148	.0792753	3.49	0.000	.121038 .4317915
age2	-.0018315	.0005265	-3.48	0.001	-.0028633 -.0007996
actlim	.357446	.0454683	7.86	0.000	.2683298 .4465622
totchr	.4035182	.0162768	24.79	0.000	.3716163 .4354201
medicaid	.0893386	.0622201	1.44	0.151	-.0326105 .2112878
private	.0775393	.0442757	1.75	0.080	-.0092395 .1643181
_cons	-4.402228	2.969646	-1.48	0.138	-10.22263 1.418171
DRUG_lnvar					
_cons	.2695824	.0309126	8.72	0.000	.2089948 .3301699
OTHER_mean					
age	.3173817	.1029119	3.08	0.002	.115678 .5190854
age2	-.0020875	.0006844	-3.05	0.002	-.0034289 -.0007461
educyr	.0650207	.0076146	8.54	0.000	.0500964 .079945
actlim	.7421208	.0635696	11.67	0.000	.6175266 .866715
totchr	.295988	.0204506	14.47	0.000	.2559057 .3360704
private	.258998	.0542141	4.78	0.000	.1527402 .3652557
_cons	-6.1414	3.849682	-1.60	0.111	-13.68664 1.403837
OTHER_lnvar					
_cons	.7906974	.0241107	32.79	0.000	.7434413 .8379534

```

. test _b[DRUG_mean:private] = _b[OTHER_mean:private]
( 1) [DRUG_mean]private - [OTHER_mean]private = 0
      chi2( 1) =    7.93
      Prob > chi2 =  0.0049

```

The default for the `suest` command is to report heteroskedastic robust standard errors. The command `suest DRUG OTHER, vce(cluster age)` would instead produce cluster-robust standard errors and subsequent tests, with clustering on `age`.

## 6.9 Survey data: Weighting, clustering, and stratification

We now turn to a quite different topic: adjustments to standard estimation methods when the data are not from a simple random sample, as we have implicitly assumed, but instead come from complex survey data. The issues raised apply to all estimation methods, including single-equation least-squares estimation of the linear model, on which we focus here.

Complex survey data lead to a sample that can be weighted, clustered, and stratified. From section [3.8](#), weighted estimation, if desired, can be performed by using the estimation command modifier `[pweight=weight]`. (This is a quite different reason for weighting than is that leading to the use of `aweights` in section [6.3.4](#).) Valid standard errors that control for clustering can be obtained by using the `vce(cluster clustvar)` option. This is the usual approach in microeconometric analysis—standard errors should always control for any clustering of errors, and weighted analysis may or may not be appropriate depending on whether a census coefficient approach or a model approach is taken; see section [3.8.3](#).

The drawback to this approach is that while it yields valid estimates, it ignores the improvement in precision of these estimates that arises because of stratification. This leads to conservative inference that uses overestimates of the standard errors, though for regression analysis, this overestimation need not be too large. The attraction of survey commands, performed in Stata by using the `svy` prefix, is that they simultaneously do all three adjustments, including that for stratification.

### 6.9.1 Survey design

As an example of using complex survey data, we begin with a modified version of the `nhanes2.dta`, which is provided at the Stata website. These data come from the second National Health and Nutrition Examination Survey, a U.S. survey conducted in 1976–1980.

We consider models for the hemoglobin count, a measure of the amount of the oxygen-transporting protein hemoglobin present in one's blood. Low

values are associated with anemia. We estimate both the mean and the relationship with age and gender, restricting analysis to nonelderly adults. The question being asked is a purely descriptive one of how does hemoglobin vary with age and gender in the population. To answer the question, we should use sampling weights because the sample design is such that different types of individuals appear in the survey with different probabilities.

Here is a brief explanation of the survey design for the data analyzed: The country is split into 32 geographical strata. Each stratum contains a number of primary sampling units (PSUs), where a PSU represents a county or several contiguous counties with an average population of several hundred thousand people. Exactly 2 PSUs were chosen from each of the 32 strata, and then several hundred individuals were sampled from each PSU. The sampling of PSUs and individuals within the PSU was not purely random, so sampling weights are provided to enable correct estimation of population means at the national level. Observations on individuals may be correlated within a given PSU but are uncorrelated across PSUs, so there is clustering on the PSU. And the strata are defined so that PSUs are more similar within strata than they are across strata. This stratification improves estimator efficiency.

We can see descriptions and summary statistics for the key survey design variables and key analysis variables by typing

```

. * Survey data example: NHANES II data
. clear all
. qui use mus206nhanes
. qui keep if age >= 21 & age <= 65
. describe samp1 finalwgt strata psu

```

Variable name	Storage type	Display format	Value label	Variable label
samp1	long	%9.0g		Unique case identifier
finalwgt	long	%9.0g		Sampling weight (except lead)
strata	byte	%9.0g		Stratum identifier, 1-32
psu	byte	%9.0g		Primary sampling unit, 1 or 2

```

. summarize samp1 finalwgt strata psu

```

Variable	Obs	Mean	Std. dev.	Min	Max
samp1	8,136	33518.94	18447.04	1400	64702
finalwgt	8,136	12654.81	7400.205	2079	79634
strata	8,136	16.67146	9.431087	1	32
psu	8,136	1.487955	.4998856	1	2

There are three key survey design variables. The sample weights are given in `finalwgt` and take on a wide range of values, so weighting may be important. The strata are given in `strata` and are numbered 1 to 32. The `psu` variable defines each PSU within strata and takes on only values of 1 and 2 because there are two PSUs per strata.

Before survey commands can be used, the survey design must be declared by using the `svyset` command. For a single-stage survey, the command syntax is

```
svyset [psu] [weight] [, design_options options]
```

For our data, we can provide all three of these quantities, as follows:

```

. * Declare survey design
. svyset psu [pweight=finalwgt], strata(strata)

Sampling weights: finalwgt
      VCE: linearized
      Single unit: missing
      Strata 1: strata
Sampling unit 1: psu
      FPC 1: <zero>

```

For our dataset, the PSU variable was named `psu`, and the strata variable was named `strata`, but other names could have been used. The output

`vce: linearized` means the VCE will be estimated using Taylor linearization, which is analogous to cluster-robust methods in the nonsurvey case. An alternative that we do not consider is balanced repeated replication, which can be an improvement on linearization and requires provision of replicate-weight variables that ensure respondent confidentiality, whereas provision of variables for the strata and PSU may not. The output `FPC 1: <zero>` means that no finite-population correction (FPC) is provided. The FPC corrects for the complication that sampling is without replacement rather than with replacement, but this correction is necessary only if a considerable portion of the PSUs in a stratum are actually sampled. The FPC is generally unnecessary for a national survey of individuals, unless the number of PSUs in a stratum is very small.

The design information is given for a single-stage survey. In fact, the second National Health and Nutrition Examination Survey is a multistage survey with sample segments (usually city blocks) chosen from within each PSU, households chosen from within each segment, and individuals chosen from within each household. This additional information can also be provided in `svyset` but is often not available for confidentiality reasons, and by far the most important information is declaring the first-stage sampling units.

The `svydescribe` command gives details on the survey design:

```

. * Describe the survey design
. svydescribe

Survey: Describing stage 1 sampling units

Sampling weights: finalwgt
      VCE: linearized
      Single unit: missing
      Strata 1: strata
Sampling unit 1: psu
      FPC 1: <zero>

```

Stratum	# units	# obs	Number of obs per unit		
			Min	Mean	Max
1	2	286	132	143.0	154
2	2	138	57	69.0	81
3	2	255	103	127.5	152
4	2	369	179	184.5	190
5	2	215	93	107.5	122
6	2	245	112	122.5	133
7	2	349	145	174.5	204
8	2	250	114	125.0	136
9	2	203	88	101.5	115
10	2	205	97	102.5	108
11	2	226	105	113.0	121
12	2	253	123	126.5	130
13	2	276	121	138.0	155
14	2	327	163	163.5	164
15	2	295	145	147.5	150
16	2	268	128	134.0	140
17	2	321	142	160.5	179
18	2	287	117	143.5	170
20	2	221	95	110.5	126
21	2	170	84	85.0	86
22	2	242	98	121.0	144
23	2	277	136	138.5	141
24	2	339	162	169.5	177
25	2	210	94	105.0	116
26	2	210	103	105.0	107
27	2	230	110	115.0	120
28	2	229	106	114.5	123
29	2	351	165	175.5	186
30	2	291	134	145.5	157
31	2	251	115	125.5	136
32	2	347	166	173.5	181
31	62	8,136	57	131.2	204

For this data extract, only 31 of the 32 strata are included (stratum 19 is excluded), and each stratum has exactly 2 PSUs, so there are 62 distinct PSUs in all.

### 6.9.2 Survey mean estimation

We consider estimation of the population mean of `hgb`, the hemoglobin count with a normal range of approximately 12–15 for women and 13.5–16.5 for men. To estimate the population mean, we should definitely use the sampling weights.

To additionally control for clustering and stratification, we give the `svy` prefix before `mean`. We have

Survey: Mean estimation				
	Linearized			
	Mean	std. err.	[95% conf. interval]	
Number of strata	= 31		Number of obs =	8,136
Number of PSUs	= 62		Population size =	102,959,526
			Design df =	31
hgb	14.29713	.0345366	14.22669	14.36757

The population mean is quite precisely estimated with a 95% confidence interval [14.23, 14.37].

What if we completely ignored the survey design? We have

Mean estimation		Number of obs = 8,136		
		Mean	Std. err.	[95% conf. interval]
hgb		14.28575	.0153361	14.25569 14.31582

In this example, the estimate of the population mean is essentially unchanged. There is a big difference in the standard errors. The default standard-error estimate of 0.015 is wrong for two reasons: it is underestimated because of failure to control for clustering, and it is overestimated because of failure to control for stratification. Here  $0.015 < 0.035$ , so, as in many cases, the failure to control for clustering dominates and leads to great overstatement of the precision of the estimator.

### 6.9.3 Survey linear regression

The `svy` prefix before `regress` simultaneously controls for weighting, clustering, and stratification declared in the preceding `svyset` command. We type

```
. * Regression using svy:  
. svy: regress hgb age female  
(running regress on estimation sample)  
  
Survey: Linear regression  
  
Number of strata = 31  
Number of PSUs    = 62  
  
Number of obs     =      8,136  
Population size  = 102,959,526  
Design df        =       31  
F(2, 30)          =    2071.57  
Prob > F         =    0.0000  
R-squared         =    0.3739
```

hgb	Linearized					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
age	.0021623	.0010488	2.06	0.048	.0000232	.0043014
female	-1.696847	.0261232	-64.96	0.000	-1.750125	-1.643568
_cons	15.0851	.0651976	231.38	0.000	14.95213	15.21807

The hemoglobin count increases slightly with age and is considerably lower for women when compared with the sample mean of 14.3.

The same weighted estimates, with standard errors that control for clustering but not stratification, can be obtained without using survey commands. To do so, we first need to define a single variable that uniquely identifies each PSU, whereas survey commands can use two separate variables, here `strata` and `psu`, to uniquely identify each PSU. Specifically, `strata` took 31 different integer values, while `psu` took only the values 1 and

2. To make 62 unique PSU identifiers, we multiply `strata` by 2 and add `psu`. Then we have

```
. * Regression using weights and cluster on PSU
. generate uniqpsu = 2*strata + psu // Make unique identifier for each PSU
. regress hgb age female [pweight=finalwgt], vce(cluster uniqpsu)
(sum of wgt is 102,959,526)

Linear regression
Number of obs      =     8,136
F(2, 61)          =   1450.50
Prob > F          =    0.0000
R-squared          =    0.3739
Root MSE           =    1.0977

(Std. err. adjusted for 62 clusters in uniqpsu)
```

hgb	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
age	.0021623	.0011106	1.95	0.056	-.0000585	.0043831
female	-1.696847	.0317958	-53.37	0.000	-1.760426	-1.633267
_cons	15.0851	.0654031	230.65	0.000	14.95432	15.21588

The regression coefficients are the same as before. The standard errors for the slope coefficients are roughly 5% and 20% larger than those obtained when the `svy` prefix is used, so using survey methods to additionally control for stratification improves estimator efficiency.

Finally, consider a naive OLS regression without weighting or without obtaining cluster-robust VCE:

```
. * Regression using no weights and no cluster
. regress hgb age female

Source | SS          df          MS          Number of obs      =     8,136
        | F(2, 8133)   =   2135.79
Model  | 5360.48245   2   2680.24123  Prob > F          =    0.0000
Residual | 10206.2566  8,133  1.25491905 R-squared          =    0.3444
Total   | 15566.7391  8,135  1.91355121 Adj R-squared      =    0.3442
                    Root MSE       =    1.1202

hgb | Coefficient  Std. err.      t      P>|t|  [95% conf. interval]
     | .0013372   .0008469    1.58    0.114  -.0003231   .0029974
     | -1.624161   .024857   -65.34   0.000  -1.672887  -1.575435
     | 15.07118   .0406259   370.97   0.000  14.99154   15.15081
```

Now the coefficient of `age` has changed considerably and standard errors are, erroneously, considerably smaller because of failure to control for clustering on the PSU.

For most microeconometric analyses, one should always obtain standard errors that control for clustering, if clustering is present. Many data extracts from complex survey datasets do not include data on the PSU, for confidentiality reasons or because the researcher did not extract the variable. Then a conservative approach is to use nonsurvey methods and obtain standard errors that cluster on a variable that subsumes the PSUs, for example, a geographic region such as a state.

As emphasized in section [3.8](#), the issue of whether to weight in regression analysis (rather than mean estimation) with complex survey data is a subtle one. For the many microeconometrics applications that are assumed to include appropriate control variables, it is unnecessary.

## 6.10 Additional resources

Microeconometrics studies increasingly base inference on cluster–robust standard errors, both in cross-sectional settings similar to those studied in this chapter and in the panel setting of chapter 8. [Cameron and Miller \(2015\)](#), [MacKinnon, Nielsen, and Webb \(Forthcoming\)](#), and [MacKinnon and Webb \(2020\)](#) provide surveys of the subject. While this chapter focused on the OLS estimator, the methods generalize to nonlinear estimators such as the logit estimator and to generalized methods of moments estimators.

The `sureg` command introduces multiequation regression. Related multiequation commands are [MV] `mvreg`, [R] `nlsur`, and [R] `reg3`. The multivariate regression command `mvreg` is essentially the same as `sureg`. The `nlsur` command generalizes `sureg` to nonlinear equations; see section 18.11. The `reg3` command generalizes the SUR model to handle endogenous regressors; see section 7.10. Linear mixed models are detailed in [ME] `mixed`.

Econometrics texts give little coverage of survey methods, and the survey literature is a stand-alone literature that is relatively inaccessible to econometricians. The Stata [SVY] *Stata Survey Data Reference Manual* is quite helpful. Econometrics references include [Bhattacharya \(2005\)](#), [Cameron and Trivedi \(2005\)](#), and [Kreuter and Valliant \(2007\)](#). [Abadie et al. \(2022\)](#) propose inference that is both sampling based and designed based; see section 24.4.7.

## 6.11 Exercises

1. Generate data by using the same DGP as that in section [6.3](#), and implement the first step of FGLS estimation to get the predicted variance `varu`. Now, compare several different methods to implement the second step of weighted estimation. First, use `regress` with the modifier `[aweight=1/varu]`, as in the text. Second, manually implement this regression by generating the transformed variable `try=y/sqrt(varu)` and regressing `try` on the similarly constructed variables `trx2`, `trx3`, and `trone`, using `regress` with the `noconstant` option. Third, use `regress` with `[pweight=1/varu]`, and show that the default standard errors using `pweights` differ from those using `aweights` because the `pweights` default is to compute robust standard errors.
2. Consider the same DGP as that in section [6.3](#). Given this specification of the model, the rescaled equation
$$y/w = \beta_1(1/w) + \beta_2(x_2/w) + \beta_3(x_3/w) + e,$$
where
$$w = \sqrt{\exp(-1 + 0.2x_2)}$$
will have the error  $e$ , which is normally distributed and homoskedastic. Treat  $w$  as known, and estimate this rescaled regression in Stata by using `regress` with the `noconstant` option. Compare the results with those given in section [6.3](#), where the weight  $w$  was estimated. Is there a big difference here between the GLS and FGLS estimators?
3. Consider the same DGP as that in section [6.3](#). Suppose that we incorrectly assume that  $u \sim N(0, \sigma^2 x_2^2)$ . Then FGLS estimates can be obtained by using `regress` with `[pweight=1/x2sq]`, where `x2sq=x2^2`. How different are the estimates of  $(\beta_1, \beta_2, \beta_3)$  from the OLS results? Can you explain what has happened in terms of the consequences of using the wrong skedasticity function? Do the standard errors change much if robust standard errors are computed? Use the `estat hettest` command to check whether the regression errors in the transformed model are homoskedastic.
4. For the model and data of section [6.7](#), verify that `mixed` with the `mle` option gives the same results as `xtreg, mle`. Also, compare the results with those from using `mixed` with the `reml` option. Fit the two-way RE model assuming random individual and time effects, and compare

results with those from when the time effects are allowed to be fixed (in which case time dummies are included as regressors).

5. Consider the same dataset as in section [6.8](#). Repeat the analysis of section [6.8](#) using the dependent variables `drugexp` and `totothr`, which are in levels rather than logs (so heteroskedasticity is more likely to be a problem). First, estimate the two equations using OLS with default standard errors and robust standard errors, and compare the standard errors. Second, estimate the two equations jointly using `sureg`, and compare the estimates with those from OLS. Third, use the `bootstrap` prefix to obtain robust standard errors from `sureg`, and compare the efficiency of joint estimation with that of OLS. Hint: It is much easier to compare estimates across methods if the `estimates` command is used; see section [3.5.6](#).
6. Consider the same dataset as in section [6.9](#). Repeat the analysis of section [6.9](#) using all observations rather than restricting the sample to ages between 21 and 65 years. First, declare the survey design. Second, compare the unweighted mean of `hgb` and its standard error, ignoring survey design, with the weighted mean and standard error, allowing for all features of the survey design. Third, do a similar comparison for least-squares regression of `hgb` on `age` and `female`. Fourth, estimate this same regression using `regress` with `pweights` and cluster-robust standard errors, and compare with the survey results.
7. Reconsider the dataset from section [6.8.3](#). Estimate the parameters of each equation by OLS. Compare these OLS results with the SUR results reported in section [6.8.3](#).



# **Chapter 7**

## **Linear instrumental-variables regression**

## 7.1 Introduction

The fundamental assumption for consistency of least-squares estimators is that the model error term is not correlated with the regressors; that is,  $\text{Cov}(u, \mathbf{x}) = \mathbf{0}$ .

If this assumption fails, the ordinary least-squares (OLS) estimator is inconsistent, and the OLS estimator can no longer be given a causal interpretation. Specifically, the OLS estimate  $\hat{\beta}_j$  can no longer be interpreted as estimating the marginal effect on the dependent variable  $y$  of an exogenous change in the  $j$ th regressor variable  $x_j$ . This is a fundamental problem because such marginal effects are a key input to economic policy.

The instrumental-variables (IV) estimator provides a consistent estimator under the very strong assumption that valid instruments exist, where the instruments  $\mathbf{z}$  are variables that are correlated with the regressors  $\mathbf{x}$  that satisfy  $\text{Cov}(u, \mathbf{z}) = \mathbf{0}$ .

The IV approach is the original and leading approach for estimating the parameters of models with endogenous regressors and errors-in-variables models. Mechanically, the IV method is no more difficult to implement than OLS regression.

Conceptually, the IV method is more difficult than other regression methods because it can be very difficult to obtain instruments  $\mathbf{z}$  that are valid: they must be variables that do not directly determine the dependent variable  $y$  to ensure that  $E(u|\mathbf{z}) = 0$ .

Even where such instruments exist, they may be weakly correlated with the endogenous regressors, leading to a great loss of estimator precision. And in extreme cases of weak correlation that are often encountered in practice, standard asymptotic theory provides a poor guide in finite samples. The Anderson–Rubin test provides a way to obtain valid inference on the coefficient of an endogenous regressor in the presence of weak instruments while using regular asymptotics. But this method entails substantial loss of power in overidentified models with many instruments.

Then statistical inference is obtained under weak-instrument asymptotics that are well established when model errors are independent and identically distributed (i.i.d.) but are still being developed when model errors are not i.i.d.

For a long time, the IV method has been the leading method used in microeconomics to establish estimates that can be given a causal interpretation. More recently, experimental and quasiexperimental methods have been increasingly used; these methods are detailed in chapters 24 and 25.

## 7.2 Simultaneous equations model

It is helpful to begin the topic of regression with endogenous regressors with a brief coverage of the classic simultaneous equations model because the IV method arises naturally in models consisting of two or more endogenous variables with two or more corresponding equations, one equation for each endogenous variable. For example, in the classic market-clearing demand and supply model, the two equations would correspond to the two endogenous interdependent variables, equilibrium price and quantity traded, whose values would be determined jointly by supply and demand.

### 7.2.1 Structural model

We consider a system of  $G$  equations in  $G$  variables  $\mathbf{y}$ . The complication is that these variables are interdependent. Thus, they are called endogenous, meaning that they come from within the system. Additionally, the system includes  $K$  variables  $\mathbf{z}$ , called exogenous variables, whose values are determined outside the system.

For the  $i$ th observation in the system, the  $G$  stochastic equations are written in matrix notation as

$$\mathbf{y}'_i \mathbf{B} + \mathbf{z}'_i \boldsymbol{\Gamma} = \mathbf{u}'_i \quad (7.1)$$

where  $\mathbf{y}_i$ ,  $\mathbf{B}$ ,  $\mathbf{z}_i$ ,  $\boldsymbol{\Gamma}$ , and  $\mathbf{u}_i$  have dimensions  $(G \times 1)$ ,  $(G \times G)$ ,  $(K \times 1)$ ,  $(K \times G)$ , and  $(G \times 1)$ , respectively, and  $\beta_{jk}$ , the  $jk$ th entry in  $\mathbf{B}$ , is the coefficient of  $y_{ki}$  in the equation for  $y_{ji}$ . Then, for example,  $y_{1i}$  depends on a linear combination of  $y_{2i}, \dots, y_{Gi}$ , a linear combination of  $z_{1i}, \dots, z_{Ki}$ , and the error  $u_{1i}$ .

The matrix  $\mathbf{B}$  is normalized to have 1 along the diagonal and should have rank  $G$  if the equation system is complete. The off-diagonal elements of  $\mathbf{B}$  reflect the interdependence between the endogenous variables. Special notable cases are  $\mathbf{B}$  diagonal, which implies no direct interdependence, and

$\mathbf{B}$  lower triangular, in which case direct dependence is recursive or unidirectional.

In the language of simultaneous equations, the linear model (7.1) is referred to as the structural model. Its central importance arises from its property that it purports to model the direct interdependence between endogenous variables.

Interdependence between elements of  $\mathbf{y}$  can be indirect through nonzero covariance between the elements of  $\mathbf{u}$ . A standard assumption is  $E(\mathbf{u}_i) = \mathbf{0}$  and  $E(\mathbf{u}_i \mathbf{u}'_i) = \Sigma$ , where  $\Sigma$  is a symmetric positive definite matrix. One special notable case is a diagonal  $\Sigma$ , which implies that random shocks that impact individual elements of  $\mathbf{y}$  are mutually uncorrelated.

The elements of matrices  $\mathbf{B}$  and  $\Gamma$  are subject to a priori assumptions that play an important role in identification of the model. For example, if a certain off-diagonal element of  $\mathbf{B}$  is set to 0, say  $\beta_{jk} = 0$ , then this means that a priori one rules out that  $y_k$  directly affects  $y_j$ . Similarly,  $\gamma_{jk} = 0$  means that a priori  $z_k$  does not directly impact  $y_j$ . Such a priori restrictions on dependence are necessary for unique identification of the parameters of the model.

The structural representation has a special appeal for several reasons.

1. Equations themselves have interpretations as economic relationships such as, for example, demand or supply relations.
2. Equations are grounded in economic theory in the sense that  $(\mathbf{B}, \Gamma, \Sigma)$  are subject to restrictions of economic theory.
3.  $\mathbf{B}$  embodies “causal” or direct connections between endogenous variables that are often the key target of empirical inquiry.

### 7.2.2 Reduced-form model

The equation system (7.1) is driven by changes in  $(\mathbf{z}_i, \mathbf{u}_i)$ . This can be seen directly by reexpressing the structural model in a solved form for  $\mathbf{y}_i$ . For specified values of  $(\mathbf{B}, \Gamma)$  and  $(\mathbf{z}_i, \mathbf{u}_i)$ , the  $G$  linear simultaneous equations can in principle be solved for  $\mathbf{y}_i$  by postmultiplying the structural model (7.1) by  $\mathbf{B}^{-1}$ .

Then  $\mathbf{y}'_i + \mathbf{z}'_i \boldsymbol{\Gamma} \mathbf{B}^{-1} = \mathbf{u}'_i \mathbf{B}^{-1}$ , leading to the reduced-form model

$$\mathbf{y}'_i = \mathbf{z}'_i \boldsymbol{\Pi} + \mathbf{v}'_i \quad (7.2)$$

where the reduced-form error  $\mathbf{v}'_i = \mathbf{u}'_i \mathbf{B}^{-1}$  and the reduced-form coefficients are given by

$$\boldsymbol{\Pi} = -\boldsymbol{\Gamma} \mathbf{B}^{-1} \quad (7.3)$$

In the simultaneous equations framework, the reduced form (7.2) is of interest because of the following:

1. It captures both the direct and indirect effects of change in an exogenous variable.
2. It is always identified given sufficient sample variation.
3. It permits conditional prediction of endogenous outcomes defined as  $E(\mathbf{y}|\mathbf{z}') = \mathbf{z}' \boldsymbol{\Pi}$ .
4. It also embodies the a priori restrictions on the structural specification.
5. It offers the potential of identifying the structural parameters  $\mathbf{B}$  and  $\boldsymbol{\Gamma}$  when there is a unique solution of (7.3).

When the matrix equations (7.3) can be solved uniquely for the unknown elements of  $(\mathbf{B}, \boldsymbol{\Gamma})$  in terms of the elements of  $\boldsymbol{\Pi}$ , the structural model is said to be identified. Note that consistent estimation of  $\boldsymbol{\Pi}$  is possible under quite weak conditions. But, given  $\boldsymbol{\Pi}$ , additional conditions are required to identify  $(\mathbf{B}, \boldsymbol{\Gamma})$ . A necessary condition is that the matrix  $\boldsymbol{\Pi}$  should have sufficient number of entries to yield the required number of equations. This in turn means that the model should contain a sufficient number of linearly independent exogenous variables. This condition will need to be refined further. As we do so, it becomes possible to give a more precise definition of an instrumental variable.

### 7.2.3 Recursive systems

A special case that deserves specific mention is one in which the structural equations can be ordered such that the matrix  $\mathbf{B}$  is either lower or upper triangular; this is also known as the recursive case because it can be interpreted as a case in which the endogenous variables are determined recursively, with the higher-order endogenous variables being determined by the endogenous variables with lower ordering. Recursiveness implies that there is no direct feedback from the higher-order variables to the lower-order variables. However, even in this case joint dependence between the lower- and higher-order variables can arise through correlated error terms.

Methods to control for endogeneity in nonlinear models, such as those for binary outcomes and counts that are detailed in later chapters, are almost exclusively restricted to recursive systems. In the usual case of a single endogenous regressor, a “structural” equation specifies the dependent variable  $y_1$  to depend on an endogenous variable  $y_2$  and exogenous variables, while a “first-stage” equation specifies  $y_2$  to depend on exogenous variables but, importantly, not on  $y_1$ .

#### 7.2.4 Generating a sample with simultaneous dependence

The background of the preceding section is helpful for generating samples with simultaneous dependence. Monte Carlo studies of estimators for models with endogenous regressors typically use such samples.

It is not possible to directly generate a sample from the structural model (7.1) because of the interdependence of the endogenous variables. Instead, we use the reduced form of the model.

Consider the following two-equation simultaneous equations model in which  $(y_1, y_2)$  are endogenous variables and  $(x_1, x_2)$  are exogenous variables,

$$\begin{aligned} y_{1i} &= \beta_{12}y_{2i} + \gamma_{12}x_{1i} + u_{1i} \\ y_{2i} &= \beta_{21}y_{1i} + \gamma_{21}x_{2i} + u_{2i} \end{aligned}$$

where the error terms have zero mean conditional on exogenous variables,

$$\begin{aligned}E(u_{1i}|x_{1i}, x_{2i}) &= 0 \\E(u_{2i}|x_{1i}, x_{2i}) &= 0\end{aligned}$$

where  $i = 1, \dots, N$ ;  $E(u_{1i}^2) = \sigma_1^2$ ;  $E(u_{2i}^2) = \sigma_2^2$ ; and  $E(u_{1i}u_{2i}) = \sigma_{12}$ .

Note that this model excludes some variables from each equation. Here  $x_2$  is excluded from the structural equation for  $y_1$ , and  $x_1$  is excluded from the structural equation for  $y_2$ . Exclusion restrictions such as these ensure identification of the structural parameters of this model, here  $\beta_{12}$ ,  $\beta_{21}$ ,  $\gamma_{12}$ , and  $\gamma_{21}$ .

We wish to generate a sample with  $N = 1000$ ,  $\beta_{12} = \gamma_{12} = \gamma_{21} = 1$ ,  $\beta_{21} = 0.25$ , and with  $(x_{1i}, x_{2i})$  and  $(u_{1i}, u_{2i})$  being pairs of correlated random variables. To generate data on  $y_{1i}$  and  $y_{2i}$  given the preceding setup, we need to use the reduced form of the model. After a little algebra, the reduced form for  $y_{1i}$  is

$$y_{1i} = (1 - \beta_{12}\beta_{21})^{-1} \{ \gamma_{12}x_{1i} + \beta_{12}(\gamma_{21}x_{2i} + u_{2i}) + u_{1i} \}$$

Substituting this reduced-form expression for  $y_{1i}$  in the  $y_{2i}$  structural equation yields the second reduced-form equation. Then, one makes independent bivariate draws of  $(x_{1i}, x_{2i})$  and  $(u_{1i}, u_{2i})$  and generates  $(y_{1i}, y_{2i})$  after inserting the numerical values of the structural parameters. In the example,  $(x_{1i}, x_{2i})$  are bivariate normal with correlation of 0.3, and  $(u_{1i}, u_{2i})$  are bivariate normal with correlation of 0.7.

```

. * Generate data for simultaneous equations model with b12=1; b21=c12=c21=1
. qui set obs 1000
. set seed 10101
. matrix C = (1, .7, 1)                                // Variances 1, covariance 0.7
. drawnorm u1 u2, n(1000) corr(C) cstorage(lower)      // Bivariate normal (u1, u2)
. matrix C = (1, .3, 1)
. drawnorm x1 x2, n(1000) corr(C) cstorage(lower)      // Bivariate normal (x1, x2)
. generate y1 =(1/(1-.25))*(x1 + 1*(x2+u2) + u1)    // Reduced form for y1
. generate y2 = 0.25*y1 + 1*x2 + u2                   // Generating y2 given y1
. summarize y1 y2 x1 x2 u1 u2

```

Variable	Obs	Mean	Std. dev.	Min	Max
y1	1,000	.0721602	3.30318	-12.3202	11.20666
y2	1,000	.0407909	2.169822	-8.320311	7.68481
x1	1,000	.0271982	1.006009	-3.282559	3.596714
x2	1,000	-.010926	1.003211	-2.847878	3.216181
u1	1,000	.0041711	1.013016	-4.119125	3.093716
u2	1,000	.0336769	1.005745	-3.293353	3.371781

```

. correlate y1 y2 x1 x2 u1 u2
(obs=1,000)

```

	y1	y2	x1	x2	u1	u2
y1	1.0000					
y2	0.9465	1.0000				
x1	0.5609	0.3873	1.0000			
x2	0.5151	0.6538	0.3299	1.0000		
u1	0.6763	0.5597	0.0065	-0.0485	1.0000	
u2	0.7071	0.7281	0.0459	-0.0098	0.7006	1.0000

In this example, the endogenous variables  $y_1$  and  $y_2$  are very highly correlated.

### 7.2.5 Estimation in the simultaneous equations example

We consider estimation of the first equation in the system,  $y_{1i} = \beta_{12}y_{2i} + \gamma_{12}x_{1i} + u_{1i}$ . From the previous output, Cor  $(y_{2i}, u_{1i}) = 0.56$ , so the regressor  $y_{2i}$  is highly correlated with the error term. Thus, OLS yields inconsistent parameter estimates.

We obtain

```
. * OLS is inconsistent
. regress y1 y2 x1, vce(robust) noheader
```

y1	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
y2	1.306056	.0132079	98.88	0.000	1.280137	1.331974
x1	.7508814	.0273001	27.50	0.000	.6973092	.8044537
_cons	-.0015376	.0255905	-0.06	0.952	-.051755	.0486797

The quite precise slope coefficient estimates are 1.306 and 0.751, substantially and statistically different from the data-generating process (DGP) values of 1.0 and 1.0.

The IV method explained below yields consistent estimates, given existence of a variable that does not belong in the equation for  $y_{1i}$  yet is correlated with the endogenous regressor  $y_{2i}$ . The variable  $x_{2i}$  was excluded from the  $y_{1i}$  equation, and from the preceding output,  $\text{Cor}(x_{2i}, u_i) = -0.05$  and  $\text{Cor}(y_{2i}, x_{2i}) = 0.65$ .

Using the `ivregress` command, presented in section [7.4.1](#), we obtain

```
. * IV with valid instrument (here x2 for y2) are consistent
. ivregress 2sls y1 (y2=x2) x1, vce(robust) noheader
```

y1	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
y2	.9550988	.0289822	32.95	0.000	.8982947	1.011903
x1	1.04403	.0399915	26.11	0.000	.9656476	1.122412
_cons	.0048051	.0337553	0.14	0.887	-.061354	.0709642

Instrumented: y2

Instruments: x1 x2

The slope coefficient estimates are 0.955 and 1.044 and at level 0.05 are not statistically different from the DGP values of 1.0 and 1.0.

Similarly, OLS of  $y_2$  on  $y_1$  and  $x_2$  yields inconsistent estimates, and instead we should perform IV regression with  $x_1$  an instrument for  $y_1$ .

Consistent estimates by joint estimation of the entire system can be obtained using the `reg3` command presented in section [7.10](#). In this special

example, the results are identical to those obtained by separate IV estimation of each equation.

## 7.3 Instrumental-variables regression

IV estimation enables estimation of a single equation in a system. Furthermore, it does not require specification of the remaining equations in a structural system. It only requires specification of one or more instrumental variables.

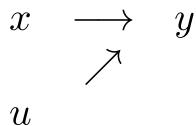
### 7.3.1 Basic IV theory

We introduce IV methods in the simplest regression model, one where the dependent variable  $y$  is regressed on a single regressor  $x$ :

$$y = \beta x + u \quad (7.4)$$

The model is written without the intercept. This leads to no loss of generality if both  $y$  and  $x$  are measured as deviations from their respective means.

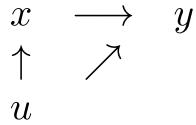
For concreteness, suppose  $y$  measures earnings,  $x$  measures years of schooling, and  $u$  is the error term. The simple regression model assumes that  $x$  is uncorrelated with the errors in (7.4). Then the only effect of  $x$  on  $y$  is a direct effect via the term  $\beta x$ . Schematically, we have the following path diagram:



The absence of a directional arrow from  $u$  to  $x$  means that there is no association between  $x$  and  $u$ . Then the OLS estimator  $\hat{\beta} = \sum_i x_i y_i / \sum_i x_i^2$  is consistent for  $\beta$ .

The error  $u$  embodies all factors other than schooling that determine earnings. One such factor in  $u$  is ability. However, high ability will induce

correlation between  $x$  and  $u$  because high (low) ability will on average be associated with high (low) years of schooling. Then a more appropriate schematic diagram is



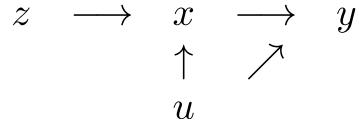
where now there is an association between  $x$  and  $u$ .

The OLS estimator  $\hat{\beta}$  is then inconsistent for  $\beta$  because  $\hat{\beta}$  combines the desired direct effect of schooling on earnings ( $\beta$ ) with the indirect effect that people with high schooling are likely to have high ability, high  $u$ , and hence high  $y$ . For example, if one more year of schooling is found to be associated on average with a \$1,000 increase in annual earnings, we are not sure how much of this increase is due to schooling per se ( $\beta$ ) and how much is due to people with higher schooling having on average higher ability (so higher  $u$ ).

The regressor  $x$  is said to be endogenous, meaning it arises within a system that influences  $u$ . By contrast, an exogenous regressor arises outside the system and is unrelated to  $u$ . The inconsistency of  $\hat{\beta}$  is referred to as endogeneity bias because the bias does not disappear asymptotically.

The obvious solution to the endogeneity problem is to include as regressors controls for ability. But such regressors may not be available. Few earnings–schooling datasets additionally have measures of ability such as IQ tests; even if they do, there are questions about the extent to which they measure inherent ability.

The IV approach provides an alternative solution. We introduce a (new) instrumental variable,  $z$ , which has the property that changes in  $z$  are associated with changes in  $x$  but do not lead to changes in  $y$  (except indirectly via  $x$ ). This leads to the following path diagram:



For example, proximity to college ( $z$ ) may determine college attendance ( $x$ ) but not directly determine earnings ( $y$ ).

The IV estimator in this simple example is  $\widehat{\beta}_{\text{IV}} = \sum_i z_i y_i / \sum_i z_i x_i$ . This can be interpreted as the ratio of the covariance of  $y$  with  $z$  to the covariance of  $x$  with  $z$  or, after some algebra, as the ratio of  $dy/dz$  to  $dx/dz$ . For example, if a one-unit increase in  $z$  is associated with 0.2 more years of education and with \$500 higher earnings, then  $\widehat{\beta}_{\text{IV}} = \$500/0.2 = \$2500$ , so one more year of schooling increases earnings by \$2,500.

The IV estimator  $\widehat{\beta}_{\text{IV}}$  is consistent for  $\beta$  provided that the instrument  $z$  is uncorrelated with the error  $u$  and correlated with the regressor  $x$ .

### 7.3.2 Model setup

We now consider the more general regression model with the scalar dependent variable  $y_1$ , which depends on  $m$  endogenous regressors, denoted by  $\mathbf{y}_2$ , and  $K_1$  exogenous regressors (including an intercept), denoted by  $\mathbf{x}_1$ . This model is called a structural equation, with

$$y_{1i} = \mathbf{y}'_{2i} \boldsymbol{\beta}_1 + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + u_i, \quad i = 1, \dots, N \quad (7.5)$$

The regression errors  $u_i$  are assumed to be uncorrelated with  $\mathbf{x}_{1i}$  but are correlated with  $\mathbf{y}_{2i}$ . This correlation leads to the OLS estimator being inconsistent for  $\boldsymbol{\beta}$ .

To obtain a consistent estimator, we assume the existence of at least  $m$  instrumental variables  $\mathbf{x}_2$  for  $\mathbf{y}_2$  that satisfy the assumption that  $E(u_i | \mathbf{x}_{2i}) = 0$ . The instruments  $\mathbf{x}_2$  need to be correlated with  $\mathbf{y}_2$  so that they provide some information on the variables being instrumented. One way to

motivate this is to assume that each component  $y_{2j}$  of  $\mathbf{y}_2$  satisfies the first-stage equation

$$y_{2ji} = \mathbf{x}'_{1i}\boldsymbol{\pi}_{1j} + \mathbf{x}'_{2i}\boldsymbol{\pi}_{2j} + v_{ji}, \quad j = 1, \dots, m \quad (7.6)$$

The first-stage equations have only exogenous variables on the right-hand side. The first-stage equation could be obtained by first specifying a simultaneous equations system for all endogenous variables, as in section 7.2.1, and thus, it is sometimes also called a reduced-form equation. But often first-stage equations are specified without appeal to an underlying simultaneous equations system.

The exogenous regressors  $\mathbf{x}_1$  in (7.5) can be used as instruments for themselves. The challenge then is to come up with additional instruments  $\mathbf{x}_2$ . Often,  $\mathbf{y}_2$  is scalar,  $m = 1$ , and it is enough to find one additional instrument  $x_2$ . More generally, with  $m$  endogenous regressors, we need at least  $m$  additional instruments  $\mathbf{x}_2$ . This can be difficult because  $\mathbf{x}_2$  needs to be a variable that can be legitimately excluded from the structural model (7.5) for  $y_1$ .

The model (7.5) can be more simply written as

$$y_i = \mathbf{x}'_i\boldsymbol{\beta} + u_i$$

where the regressor vector  $\mathbf{x}'_i = [\mathbf{y}'_{2i} \mathbf{x} \mathbf{x}'_{1i}]$  combines endogenous and exogenous variables and the dependent variable is denoted by  $y$  rather than  $y_1$ . We similarly combine the instruments for these variables. Then the vector of instrumental variables (or, more simply, instruments) is  $\mathbf{z}'_i = [\mathbf{x}'_{1i} \mathbf{x} \mathbf{x}'_{2i}]$ , where  $\mathbf{x}_1$  serves as the (ideal) instrument for itself and  $\mathbf{x}_2$  is the instrument for  $\mathbf{y}_2$  and the instruments  $\mathbf{z}$  satisfy the conditional moment restriction

$$E(u_i|\mathbf{z}_i) = 0 \quad (7.7)$$

In summary, we regress  $y$  on  $x$  using instruments  $z$ .

### 7.3.3 IV estimators: IV, two-stage least-squares, and generalized method of moments

The key (and in many cases, heroic) assumption is (7.7). This implies that  $E(z_i u_i) = 0$ , and hence the moment condition, or population zero-correlation condition,

$$E \{ z'_i (y_i - x'_i \beta) \} = 0 \quad (7.8)$$

IV estimators are solutions to the sample analog of (7.8).

We begin with the case where  $\dim(z) = \dim(x)$ , called the just-identified case, where the number of instruments exactly equals the number of regressors. Then the sample analog of (7.8) is  $\sum_{i=1}^N z'_i (y_i - x'_i \beta) = 0$ . As usual, stack the vectors  $x'_i$  into the matrix  $X$ , the scalars  $y_i$  into the vector  $y$ , and the vectors  $z'_i$  into the matrix  $Z$ . Then we have  $Z'(y - X\beta) = 0$ . Solving for  $\beta$  leads to the IV estimator

$$\hat{\beta}_{IV} = (Z'X)^{-1}Z'y$$

A second case is where  $\dim(z) < \dim(x)$ , called the not-identified or underidentified case, where there are fewer instruments than regressors. Then no consistent IV estimator exists. This situation often arises in practice. Obtaining enough instruments, even just one in applications with a single endogenous regressor, can require considerable ingenuity or access to unusually rich data.

A third case is where  $\dim(z) > \dim(x)$ , called the overidentified case, where there are more instruments than regressors. This can happen especially when economic theory leads to clear exclusion of variables from the equation of interest, freeing up these variables to be used as instruments if they are correlated with the included endogenous regressors. Then

$\mathbf{Z}'(\mathbf{y} - \mathbf{X}\beta) = \mathbf{0}$  has no solution for  $\beta$  because it is a system of  $\dim(\mathbf{z})$  equations in only  $\dim(\mathbf{x})$  unknowns. One possibility is to arbitrarily drop instruments to get to the just-identified case. But there are more efficient estimators. One estimator is the two-stage least-squares (2SLS) estimator,

$$\hat{\beta}_{\text{2SLS}} = \left\{ \mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X} \right\}^{-1} \mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$$

This estimator is the most efficient estimator if the errors  $u_i$  are independent and homoskedastic. And it equals  $\hat{\beta}_{\text{IV}}$  in the just-identified case.

The term “2SLS” arises because the estimator can be computed in two steps. First, estimate by OLS the first-stage regressions given in (7.6), and second, estimate by OLS the structural regression (7.5) with endogenous regressors replaced by predictions from the first step. This mechanical interpretation is peculiar to linear models and does not generalize to nonlinear regression models; see, for example, section 20.7.

A quite general estimator is the linear generalized method of moments (GMM) estimator

$$\hat{\beta}_{\text{GMM}} = (\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{y} \quad (7.9)$$

where  $\mathbf{W}$  is any full-rank symmetric-weighting matrix. In general, the weights in  $\mathbf{W}$  may depend both on data and on unknown parameters. For just-identified models, all choices of  $\mathbf{W}$  lead to the same estimator. This estimator minimizes the objective function

$$Q(\beta) = \left\{ \frac{1}{N}(\mathbf{y} - \mathbf{X}\beta)' \mathbf{Z} \right\} \mathbf{W} \left\{ \frac{1}{N} \mathbf{Z}'(\mathbf{y} - \mathbf{X}\beta) \right\} \quad (7.10)$$

which is a matrix-weighted quadratic form in  $\mathbf{Z}'(\mathbf{y} - \mathbf{X}\beta)$ .

For GMM, some choices of  $\mathbf{W}$  are better than others. The 2SLS estimator is obtained with weighting matrix  $\mathbf{W} = (\mathbf{Z}'\mathbf{Z})^{-1}$ . The optimal GMM estimator uses  $\mathbf{W} = \widehat{\mathbf{S}}^{-1}$ , so

$$\widehat{\boldsymbol{\beta}}_{\text{OGMM}} = \left( \mathbf{X}' \mathbf{Z} \widehat{\mathbf{S}}^{-1} \mathbf{Z}' \mathbf{X} \right)^{-1} \mathbf{X}' \mathbf{Z} \widehat{\mathbf{S}}^{-1} \mathbf{Z}' \mathbf{y}$$

where  $\widehat{\mathbf{S}}$  is an estimate of  $\text{Var}(N^{-1/2} \mathbf{Z}' \mathbf{u})$ . If the errors  $u_i$  are independent and heteroskedastic, then  $\widehat{\mathbf{S}} = 1/N \sum_{i=1}^N \widehat{u}_i^2 \mathbf{z}_i \mathbf{z}_i'$ , where  $\widehat{u}_i = y_i - \mathbf{x}_i' \widehat{\boldsymbol{\beta}}$  and  $\widehat{\boldsymbol{\beta}}$  is a consistent estimator, usually  $\widehat{\boldsymbol{\beta}}_{\text{2SLS}}$ . The estimator reduces to  $\widehat{\boldsymbol{\beta}}_{\text{IV}}$  in the just-identified case.

In later sections, we consider additional estimators. In particular, the limited-information maximum-likelihood (LIML) estimator, while asymptotically equivalent to 2SLS, has been found in research to generally outperform both 2SLS and GMM in finite samples, though this better performance is not guaranteed.

### 7.3.4 Instrument validity and relevance

All the preceding estimators have the same starting point. The instruments must satisfy condition (7.7). This condition is impossible to test in the just-identified case. And even in the overidentified case, where a test is possible (see section 7.4.8), instrument validity relies more on persuasive argument, economic theory, and norms established in prior related empirical studies.

Additionally, the instruments must be relevant. For the model of section 7.3.2, this means that after controlling for the remaining exogenous regressors  $\mathbf{x}_1$ , the instruments  $\mathbf{x}_2$  must account for significant variation in  $y_2$ . Intuitively, the stronger the association between the instruments  $\mathbf{z}$  and  $\mathbf{x}$ , the stronger will be the identification of the model. Conversely, instruments that are only marginally relevant are referred to as weak instruments.

### 7.3.5 Weak instruments

The first consequence of an instrument being weak is that estimation becomes much less precise, so standard errors can become many times larger, and  $t$  statistics many times smaller, compared with those from (inconsistent) OLS. Then a promising  $t$  statistic of 5 from OLS estimation may become a  $t$  statistic of 1 from IV estimation. If this loss of precision is critical, then one needs to obtain better instruments or more data.

The second consequence is that even if an IV estimator is consistent, the standard asymptotic theory may provide a poor approximation to the actual sampling distribution of the IV estimator in typical finite-sample sizes. For example, the asymptotic critical values of standard Wald tests can lead to tests whose actual size differs considerably from the nominal size, and hence the tests are potentially misleading.

This problem arises in part because in finite samples the IV estimator is not centered on  $\beta$ , even though in infinite samples it is consistent for  $\beta$ . And it arises in part because the distribution is not normal in finite samples. The problem is referred to as the finite-sample bias of IV, even in situations where formally the mean of the estimator does not exist. The question “How large of a sample size does one need before these biases become unimportant?” does not have a simple answer. This issue is considered further in sections [7.5–7.8](#).

### 7.3.6 Robust standard-error estimates

Table [7.1](#) provides a summary of three leading variants of the IV family of estimators. For just-identified models, we use the IV estimator because the other models collapse to the IV estimator in that case. For overidentified models, the standard estimators are 2SLS and optimal GMM.

**Table 7.1.** IV estimators and their asymptotic variances

Estimator	Estimator definition and estimate of the VCE
IV (just-identified)	$\hat{\beta}_{\text{IV}} = (\mathbf{Z}'\mathbf{X})^{-1}\mathbf{Z}'\mathbf{y}$ $\hat{V}(\hat{\beta}) = (\mathbf{Z}'\mathbf{X})^{-1}\hat{\mathbf{S}}(\mathbf{Z}'\mathbf{X})^{-1}$
2SLS	$\hat{\beta}_{\text{2SLS}} = \{\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}\}^{-1}\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$ $\hat{V}(\hat{\beta}) = \{\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}\}^{-1}\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\hat{\mathbf{S}}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}$ $\quad \times \{\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}\}^{-1}$
Optimal GMM	$\hat{\beta}_{\text{OGMM}} = \left(\mathbf{X}'\mathbf{Z}\hat{\mathbf{S}}^{-1}\mathbf{Z}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}\hat{\mathbf{S}}^{-1}\mathbf{Z}'\mathbf{y}$ $\hat{V}(\hat{\beta}_{\text{OGMM}}) = \left(\mathbf{X}'\mathbf{Z}\hat{\mathbf{S}}^{-1}\mathbf{Z}'\mathbf{X}\right)^{-1}$

The formulas given for estimates of the VCES are robust estimates, where in this table  $\hat{\mathbf{S}}$  is an estimate of the asymptotic variance of  $\mathbf{Z}'\mathbf{u}$ .

For heteroskedastic errors,  $\hat{\mathbf{S}} = \sum_{i=1}^N \hat{u}_i^2 \mathbf{z}_i \mathbf{z}_i'$ , and we use the `vce(robust)` option.

For clustered errors, we use the `vce(cluster clustvar)` option. In that case, with  $G$  clusters,  $\hat{\mathbf{S}} = \{G/(G-1)\} \times \sum_g \mathbf{Z}_g \hat{\mathbf{u}}_g \hat{\mathbf{u}}_g' \mathbf{Z}_g'$ , where  $g$  denotes the  $g$ th cluster.

Other estimates, notably heteroskedasticity- and autocorrelation-consistent (HAC) robust variance estimates are also possible.

Sections [7.5–7.7](#) consider alternative inference methods when instruments are weak. Current research is extending methods initially developed for i.i.d. errors to non-i.i.d. errors such as heteroskedastic and clustered errors.

## 7.4 Instrumental-variables example

Most estimators in this chapter are obtained using the `ivregress` command. The rest of this section provides application to an example with a single endogenous regressor.

### 7.4.1 The `ivregress` command

The `ivregress` command performs IV regression and yields goodness-of-fit statistics, coefficient estimates, standard errors,  $t$  statistics,  $p$ -values, and confidence intervals. The syntax of the command is

```
ivregress estimator depvar [ varlist1 ] (varlist2 = varlist_iv) [ if ] [ in ] [ weight ] [ , options ]
```

Here *estimator* is one of `2sls` (2SLS), `gmm` (optimal GMM), or `liml` (LIML); *depvar* is the scalar dependent variable; *varlist1* is the list of exogenous regressors; *varlist2* is the list of endogenous regressors; and *varlist\_iv* is the list of instruments for the endogenous regressors. Note that endogenous regressors and their instruments appear inside parentheses. If the model has several endogenous variables, they are all listed on the left-hand side of the equality. Because there is no `iv` option for *estimator*, in the just-identified case, we use the `2sls` option because 2SLS is equivalent to IV in that case.

An example of the command is `ivregress 2sls y x1 x2 (y2 y3 = x3 x4 x5)`. This performs 2SLS estimation of a structural-equation model with the dependent variable, *y*; two exogenous regressors, *x1* and *x2*; two endogenous regressors, *y2* and *y3*; and three instruments, *x3*, *x4*, and *x5*. The model is overidentified because there is one more instrument than there are endogenous regressors.

In terms of the model of section 7.3.2, *y1* is *depvar*, *x1* is *varlist1*, *y2* is *varlist2*, and *x2* is *varlist\_iv*. In the just-identified case, *varlist2* and *varlist\_iv* have the same number of variables, and we use the `2sls` option to obtain the IV estimator. In the overidentified case, *varlist\_iv* has more variables than does *varlist2*.

The `first` option yields considerable output from the first-stage regression. Several useful tests regarding the instruments and the goodness of fit of the first-stage regression are displayed; therefore, this option is more convenient than the user running the first-stage regression and conducting tests.

The `vce(vcetype)` option specifies the type of standard errors reported by Stata. The options for `vcetype` are `robust`, which yields heteroskedasticity-robust standard errors; `unadjusted`, which yields nonrobust standard errors; `cluster clustvar`; `bootstrap`; `jackknife`; and `hac kernel`. Various specification test statistics that are automatically produced by Stata are made more robust if the `vce(robust)` option is used.

For the overidentified models fit by GMM, the `wmatrix(wmtype)` option specifies the type of weighting matrix used in the objective function [see **W** in (7.9)] to obtain optimal GMM. Different choices of `wmtype` lead to different estimators. For heteroskedastic errors, set `wmtype` to `robust`. For correlation between elements of a cluster, set `wmtype` to `cluster clustvar`, where `clustvar` specifies the variable that identifies the cluster. For time-series data with HAC errors, set `wmtype` to `hac kernel` or `hac kernel #` or `hac kernel opt`; see [R] **ivregress** for additional details. If `vce()` is not specified when `wmatrix()` is specified, then `vcetype` is set to `wmtype`. The `igmm` option yields an iterated version of the GMM estimator.

#### 7.4.2 Data and data summary

We consider a model with one endogenous regressor, several exogenous regressors, and one or more excluded exogenous variables that serve as the identifying instruments.

The dataset is an extract from the Medical Expenditure Panel Survey of individuals over the age of 65 years, similar to the dataset described in section 3.2.1. The equation to be estimated has the dependent variable `ldrugexp`, the log of total out-of-pocket expenditures on prescribed medications. The regressors are an indicator for whether the individual holds either employer- or union-sponsored health insurance (`hi_empunion`), number of chronic conditions (`totchr`), and four sociodemographic variables: age in years (`age`), indicators for whether female (`female`) and

whether black or Hispanic (`blhisp`), and the natural logarithm of annual household income in thousands of dollars (`linc`).

We treat the health insurance variable `hi_empunion` as endogenous. The intuitive justification is that having such supplementary insurance on top of the near universal Medicare insurance for the elderly may be a choice variable. Even though most individuals in the sample are no longer working, those who expected high future medical expenses might have been more likely to choose a job when they were working that would provide supplementary health insurance upon retirement. Note that Medicare did not cover drug expenses for the time period we study.

We use the global macro `x2list` to store the names of the variables that are treated as exogenous regressors. We have

```
. * Read data, define global x2list, and summarize data
. clear all
. qui use mus207mepspresdrugs
. global x2list totchr age female blhisp linc
. summarize ldrugexp hi_empunion $x2list
```

Variable	Obs	Mean	Std. dev.	Min	Max
ldrugexp	10,367	6.47934	1.363097	0	10.18017
hi_empunion	10,367	.3797627	.4853511	0	1
totchr	10,367	1.860712	1.290255	0	9
age	10,367	75.04128	6.688851	65	91
female	10,367	.5794347	.4936736	0	1
<hr/>					
blhisp	10,367	.1702518	.375872	0	1
linc	10,068	2.745753	.9118674	-6.907755	5.744476

Slightly less than 38% of the sample has either employer or union-sponsored health insurance in addition to Medicare insurance. Subsequent analysis drops the 299 observations with missing data on `linc`.

### 7.4.3 Available instruments

We consider four potential instruments for `hi_empunion`. Two reflect the income status of the individual, and two are based on employer characteristics.

The `ssiratio` instrument is the ratio of an individual's social security income to the individual's income from all sources, with high values indicating a significant income constraint. The `lowincome` instrument is a qualitative indicator of low-income status. Both these instruments are likely to be relevant because they are expected to be negatively correlated with having supplementary insurance. To be valid instruments, we need to assume they can be omitted from the equation for `ldrugexp`, arguing that the direct role of income is adequately captured by the regressor `linc`.

The `firmsz` instrument measures the size of the firm's employed labor force, and the `multlc` instrument indicates whether the firm is a large operator with multiple locations. These variables are intended to capture whether the individual has access to supplementary insurance through the employer. These two variables are irrelevant for those who are retired or self-employed or who purchase insurance privately. In that sense, these two instruments could potentially be weak.

```
. * Summarize available instruments
. summarize ssiratio lowincome multlc firmsz if linc!=.
```

Variable	Obs	Mean	Std. dev.	Min	Max
<code>ssiratio</code>	10,068	.5330892	.3492841	0	1
<code>lowincome</code>	10,068	.1873262	.3901925	0	1
<code>multlc</code>	10,068	.0621772	.2414891	0	1
<code>firmsz</code>	10,068	.1408224	2.172642	0	50

We have four available instruments for one endogenous regressor. The obvious approach is to use all available instruments because in theory this leads to the most efficient estimator. In practice, it may lead to larger small-sample bias because the small-sample biases of IV estimators increase with the number of instruments (Hahn and Hausman [2002](#)).

At a minimum, it is informative to use `correlate` to view the gross correlation between endogenous variables and instruments and between instruments. When multiple instruments are available, as in the case of overidentified models, then it is actually the partial correlation after controlling for other available instruments that matters. This important step is deferred to sections [7.5](#) and [7.6.5](#).

#### 7.4.4 IV estimation of an exactly identified model

We begin with iv regression of `ldrugexp` on the endogenous regressor `hi_empunion`, instrumented by the single instrument `ssiratio`, and several exogenous regressors.

We use `ivregress` with the `2sls` estimator and the options `vce(robust)` to control for heteroskedastic errors and `first` to provide output that additionally reports results from the first-stage regression. The output is in two parts:

```
. * IV estimation of a just-identified model with single endog regressor
. ivregress 2sls ldrugexp (hi_empunion = ssiratio) $x2list, vce(robust) first
First-stage regressions
```

---

Number of obs	=	10,068
F(6, 10061)	=	165.27
Prob > F	=	0.0000
R-squared	=	0.0793
Adj R-squared	=	0.0787
Root MSE	=	0.4665

---

hi_empunion	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
totchr	.0133494	.0036385	3.67	0.000	.0062172	.0204817
age	-.0084926	.0007041	-12.06	0.000	-.0098728	-.0071125
female	-.0721048	.0096379	-7.48	0.000	-.0909969	-.0532126
blhisp	-.0609939	.0122393	-4.98	0.000	-.0849853	-.0370025
linc	.0454804	.0062021	7.33	0.000	.0333231	.0576377
ssiratio	-.2205333	.0151176	-14.59	0.000	-.2501667	-.1908998
_cons	1.039144	.0575002	18.07	0.000	.9264318	1.151856

---

Instrumental variables 2SLS regression

Number of obs	=	10,068
Wald chi2(6)	=	2039.84
Prob > chi2	=	0.0000
R-squared	=	0.0831
Root MSE	=	1.3038

---

ldrugexp	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
hi_empunion	-.8115015	.1884024	-4.31	0.000	-1.180763	-.4422396
totchr	.4492135	.0100423	44.73	0.000	.429531	.468896
age	-.0122415	.0027711	-4.42	0.000	-.0176727	-.0068102
female	-.0140507	.0310339	-0.45	0.651	-.0748761	.0467747
blhisp	-.2089573	.0383529	-5.45	0.000	-.2841276	-.1337869
linc	.0815472	.0207576	3.93	0.000	.0408631	.1222314
_cons	6.692548	.2449266	27.32	0.000	6.212501	7.172595

---

Instrumented: hi\_empunion

Instruments: totchr age female blhisp linc ssiratio

The first part, added because of the `first` option, reports results from the first-stage regression of the endogenous variable `hi_empunion` on all the exogenous variables, here `ssiratio` and all the exogenous regressors in the structural equation. The first-stage regression has reasonable explanatory power, and the coefficient of `ssiratio` is negative, as expected, and highly

statistically significant. In models with more than one endogenous regressor, more than one first-stage regression is reported if the `first` option is used.

The second part reports the results of intrinsic interest, those from the IV regression of `ldrugexp` on `hi_empunion` and several exogenous regressors. Supplementary insurance has a big effect. The estimated coefficient of `hi_empunion` is  $-0.812$ , indicating that supplementary-insured individuals have out-of-pocket drug expenses that are 56% lower ( $e^{-0.812} - 1 = -0.56$ ) than those for people without employment or union-related supplementary insurance.

#### 7.4.5 IV estimation of an overidentified model

We next consider estimation of an overidentified model. Then, different estimates are obtained by 2SLS estimation and by different variants of GMM.

We use two instruments, `ssiratio` and `multlc`, for `hi_empunion`, the endogenous regressor. The first estimator is 2SLS, obtained by using `2sls` with standard errors that correct for heteroskedasticity with the `vce(robust)` option. The second estimator is optimal GMM given heteroskedastic errors, obtained by using `gmm` with the `wmatrix(robust)` option. These are the two leading estimators for overidentified IV with cross-sectional data and no clustering of the errors. The third estimator adds `igmm` to iterate to convergence. The fourth estimator is one that illustrates optimal GMM with clustered errors by clustering on `age`. The final estimator is the same as the first but reports default standard errors that do not adjust for heteroskedasticity.

```

. * Compare five estimators and variance estimates for overidentified models
. global ivmodel "ldrugexp (hi_empunion = ssiratio multlc) $x2list"
. qui ivregress 2sls $ivmodel, vce(robust)
. estimates store TwoSLS
. qui ivregress gmm $ivmodel, wmatrix(robust)
. estimates store GMM_het
. qui ivregress gmm $ivmodel, wmatrix(robust) igmm
. estimates store GMM_igmm
. qui ivregress gmm $ivmodel, wmatrix(cluster age)
. estimates store GMM_clu
. qui ivregress 2sls $ivmodel
. estimates store TwoSLS_def
. estimates table TwoSLS GMM_het GMM_igmm GMM_clu TwoSLS_def, b(%9.5f) se

```

Variable	TwoSLS	GMM_het	GMM_igmm	GMM_clu	TwoSLS_~f
hi_empunion	-0.90466 0.17962	-0.89426 0.17918	-0.89428 0.17918	-0.87513 0.16313	-0.90466 0.17821
totchr	0.45016 0.01016	0.44971 0.01014	0.44971 0.01014	0.44834 0.01359	0.45016 0.01038
age	-0.01318 0.00273	-0.01305 0.00273	-0.01305 0.00273	-0.00987 0.00612	-0.01318 0.00268
female	-0.02153 0.03095	-0.02084 0.03091	-0.02084 0.03091	-0.01282 0.02696	-0.02153 0.03035
blhisp	-0.21530 0.03863	-0.21337 0.03856	-0.21337 0.03856	-0.18363 0.04577	-0.21530 0.03805
linc	0.08891 0.02048	0.08811 0.02043	0.08811 0.02043	0.08544 0.01264	0.08891 0.02039
_cons	6.78164 0.23997	6.77124 0.23954	6.77129 0.23954	6.52887 0.48108	6.78164 0.23478

Legend: b/se

Compared with the just-identified IV estimates of section 7.4.4, the parameter estimates have changed by close to 10% (aside from those for the statistically insignificant regressor `female`). The standard errors are little changed, except for that for `hi_empunion`, which has fallen by about 5–13%, reflecting an efficiency gain due to additional instruments.

The differences between 2SLS, optimal GMM given heteroskedasticity, and iterated optimal GMM are negligible. Optimal GMM with clustering differs more. And the final column shows that the default standard errors for 2SLS

differ little from the robust standard errors in the first column, reflecting the success of the log transformation in eliminating heteroskedasticity.

#### 7.4.6 Testing for regressor endogeneity

The preceding analysis treats the insurance variable, `hi_empunion`, as endogenous. If instead the variable is exogenous, then the IV estimators (IV, 2SLS, or GMM) are still consistent, but they can be much less efficient than the OLS estimator.

The Hausman test principle provides a way to test whether a regressor is endogenous. If there is little difference between OLS and IV estimators, then there is no need to instrument, and we conclude that the regressor was exogenous. If instead there is considerable difference, then we need to instrument and the regressor was endogenous. The test usually compares just the coefficients of the endogenous variables. In the case of just one potentially endogenous regressor with a coefficient denoted by  $\beta$ , the Hausman test statistic

$$T_H = \frac{(\hat{\beta}_{\text{IV}} - \hat{\beta}_{\text{OLS}})^2}{\hat{V}(\hat{\beta}_{\text{IV}} - \hat{\beta}_{\text{OLS}})}$$

is  $\chi^2(1)$  distributed under the null hypothesis that the regressor is exogenous.

Before considering implementation of the test, we first obtain the OLS estimates to compare them with the earlier IV estimates. We have

```

. * Obtain OLS estimates to compare with preceding IV estimates
. regress ldrugexp hi_empunion $x2list, vce(robust)

Linear regression
Number of obs      =    10,068
F(6, 10061)        =     375.46
Prob > F          =     0.0000
R-squared           =     0.1769
Root MSE            =     1.2357

```

ldrugexp	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
hi_empunion	.0732427	.026008	2.82	0.005	.0222618 .1242235
totchr	.4402073	.009376	46.95	0.000	.4218285 .4585861
age	-.0033708	.0019418	-1.74	0.083	-.0071772 .0004355
female	.0569505	.0253798	2.24	0.025	.0072011 .1066999
blhisp	-.1487419	.0341819	-4.35	0.000	-.2157453 -.0817386
linc	.0116467	.0137461	0.85	0.397	-.0152985 .0385918
_cons	5.846407	.1574085	37.14	0.000	5.537855 6.154959

The OLS estimates differ substantially from the just-identified IV estimates given in section [7.4.4](#). The coefficient of `hi_empunion` has an OLS estimate of 0.073, very different from the IV estimate of  $-0.812$ . This is strong evidence that `hi_empunion` is endogenous. Some coefficients of exogenous variables also change, notably, those for `age` and `female`. Note also the loss in precision in using IV. Most notably, the standard error of the instrumented regressor increases from 0.026 for OLS to 0.188 for IV, an eightfold increase, indicating the potential loss in efficiency due to IV estimation.

The `hausman` command can be used to compute  $T_H$  under the assumption that  $\widehat{V}(\widehat{\beta}_{IV} - \widehat{\beta}_{OLS}) = \widehat{V}(\widehat{\beta}_{IV}) - \widehat{V}(\widehat{\beta}_{OLS})$ ; see section [11.9.5](#). This greatly simplifies analysis because then all that is needed are coefficient estimates and standard errors from separate IV estimation (IV, 2SLS, or GMM) and OLS estimation. But this assumption is too strong. It is correct only if  $\widehat{\beta}_{OLS}$  is the fully efficient estimator under the null hypothesis of exogeneity, an assumption that is valid only under the very strong assumption that model errors are independent and homoskedastic. One possible variation is to perform an appropriate bootstrap; see section [12.4.6](#).

The `estat endogenous` command implements the related Durbin–Wu–Hausman (DWH) test. Because the DWH test uses the device of augmented regressors, it produces a robust test statistic ([Davidson 2000](#)). The essential idea is the following. Consider the model as specified in section [7.3.1](#).

Rewrite the structural equation (7.5) with an additional variable,  $v_1$ , that is the error from the first-stage (7.6) for  $y_2$ . Then,

$$y_{1i} = \beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + \rho v_{1i} + u_i \quad (7.11)$$

Under the null hypothesis that  $y_{2i}$  is exogenous,  $E(v_{1i}u_i|y_{2i}, \mathbf{x}_{1i}) = 0$ . If  $v_1$  could be observed, then the test of exogeneity would be the test of  $H_0: \rho = 0$  in the OLS regression of  $y_1$  on  $y_2$ ,  $\mathbf{x}_1$ , and  $v_1$ . Because  $v_1$  is not directly observed, the fitted residual vector  $\hat{v}_1$  from the first-stage OLS regression (7.6) is instead substituted.

For independent homoskedastic errors, this test is asymptotically equivalent to the earlier Hausman test. In the more realistic case of errors that are heteroskedastic or clustered or have some other complication, the test of  $H_0: \rho = 0$  can still be implemented provided that we use the appropriate robust variance estimates. This test can be extended to the multiple endogenous regressors case by including multiple residual vectors and testing separately for correlation of each with the error on the structural equation.

We apply it to our example with one potentially endogenous regressor, `hi_empunion`, instrumented by `ssiratio`. Then,

```
. * Robust DWH test of endogeneity implemented by estat endogenous
. ivregress 2sls ldrugexp (hi_empunion = ssiratio) $x2list, vce(robust)

Instrumental variables 2SLS regression
                                                Number of obs      =    10,068
                                                Wald chi2(6)      =   2039.84
                                                Prob > chi2       =     0.0000
                                                R-squared          =     0.0831
                                                Root MSE           =    1.3038
```

ldrugexp	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
hi_empunion	-.8115015	.1884024	-4.31	0.000	-1.180763	-.4422396
totchr	.4492135	.0100423	44.73	0.000	.429531	.468896
age	-.0122415	.0027711	-4.42	0.000	-.0176727	-.0068102
female	-.0140507	.0310339	-0.45	0.651	-.0748761	.0467747
blhisp	-.2089573	.0383529	-5.45	0.000	-.2841276	-.1337869
linc	.0815472	.0207576	3.93	0.000	.0408631	.1222314
_cons	6.692548	.2449266	27.32	0.000	6.212501	7.172595

Instrumented: hi\_empunion

Instruments: totchr age female blhisp linc ssiratio

. estat endogenous

Tests of endogeneity

H0: Variables are exogenous

Robust score chi2(1) = 24.9812 (p = 0.0000)

Robust regression F(1,10060) = 24.8525 (p = 0.0000)

The last line of output is the robustified DWH test and leads to strong rejection of the null hypothesis that `hi_empunion` is exogenous. We conclude that it is endogenous.

We obtain exactly the same test statistic when we manually perform the robustified DWH test. We have

```
. * Robust DWH test of endogeneity implemented manually
. qui regress hi_empunion ssiratio $x2list
. qui predict v1hat, resid
. qui regress ldrugexp hi_empunion $x2list v1hat, vce(robust)
. test v1hat
( 1)  v1hat = 0
      F( 1, 10060) =    24.85
                  Prob > F =    0.0000
```

The `estat endogenous` command uses the standard errors specified in the preceding `ivregress` command. Thus, for a cluster-robust robustified

Hausman test, we use `estat endogenous` following an `ivregress`, `vce(cluster clustvar)` command.

### 7.4.7 Control function estimator

The immediately preceding OLS regression, one that adds the first-stage residuals as a regressor, remarkably leads to parameter estimates identical to the IV estimates. We have

```
. * Control function estimator adds first-stage residual as regressors
. regress ldrugexp hi_empunion $x2list v1hat, vce(robust) noheader
```

ldrugexp	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
hi_empunion	-.8115016	.1795801	-4.52	0.000	-1.163514	-.4594888
totchr	.4492135	.0094936	47.32	0.000	.4306041	.4678229
age	-.0122415	.0026367	-4.64	0.000	-.01741	-.0070729
female	-.0140507	.0293244	-0.48	0.632	-.0715323	.0434309
blhisp	-.2089573	.0364257	-5.74	0.000	-.280359	-.1375555
linc	.0815472	.0195157	4.18	0.000	.0432925	.119802
v1hat	.9039115	.1813179	4.99	0.000	.5484921	1.259331
_cons	6.692548	.2334768	28.66	0.000	6.234887	7.150209

The coefficient estimates, though not the standard errors, are identical to `ivregress` output given in section [7.4.4](#). Correct standard errors for this example that adjust for the first-stage estimation of `v1hat` are obtained in section [13.3.11](#). Alternatively, one can jointly bootstrap both steps.

This estimator, one based on [\(7.11\)](#), is called a control function estimator because the first-stage residual  $\widehat{v}_{1i}$  is added as a regressor to control for the endogenous variable  $y_{2i}$ , which is also included in the regression.

This alternative way to obtain IV estimates provides one method to control for endogeneity in some nonlinear models such as the probit and Poisson model.

### 7.4.8 Tests of overidentifying restrictions

The validity of an instrument cannot be tested in a just-identified model. But it is possible to test the validity of overidentifying instruments in an

overidentified model provided that the parameters of the model are estimated using optimal GMM. The same test has several names, including the overidentifying restrictions test, the overidentified test, Hansen's test, Sargan's test, and the Hansen–Sargan test.

The starting point is the fitted value of the criterion function (7.10) after optimal GMM; that is,

$Q(\hat{\beta}) = \{(1/N)(\mathbf{y} - \mathbf{X}\hat{\beta})'\mathbf{Z}\}\hat{\mathbf{S}}^{-1}\{(1/N)\mathbf{Z}'(\mathbf{y} - \mathbf{X}\hat{\beta})\}$ . If the population moment conditions  $E\{\mathbf{Z}'(\mathbf{y} - \mathbf{X}\beta)\} = \mathbf{0}$  are correct, then  $\mathbf{Z}'(\mathbf{y} - \mathbf{X}\hat{\beta}) \simeq \mathbf{0}$ , so  $Q(\hat{\beta})$  should be close to 0. Under the null hypothesis that all instruments are valid, it can be shown that  $Q(\hat{\beta})$  has an asymptotic chi-squared distribution with degrees of freedom equal to the number of overidentifying restrictions.

Large values of  $Q(\hat{\beta})$  lead to rejection of  $H_0: E\{\mathbf{Z}'(\mathbf{y} - \mathbf{X}\beta)\} = \mathbf{0}$ . Rejection is interpreted as indicating that at least one of the instruments is not valid. Tests can have power in other directions, however, as emphasized in section 3.7.6. It is possible that rejection of  $H_0$  indicates that the model  $\mathbf{X}\beta$  for the conditional mean is misspecified.

Going the other way, the test is one of validity only of the overidentifying instruments, so failure to reject  $H_0$  does not guarantee that all the instruments are valid. In particular, with one endogenous regressor, the test is one of whether the excess instruments are valid, conditional on the untestable assumption that at least one of the instruments is valid.

The test is implemented with the `estat overid` postestimation command following the `ivregress gmm` command for an overidentified model. We do so for the optimal GMM estimator with heteroskedastic errors and instruments, `ssiratio` and `multlc`. The example below implements `estat overid` under the overidentifying restriction.

```
. * Test of overidentifying restrictions following ivregress gmm
. qui ivregress gmm ldrugexp (hi_empunion = ssiratio multlc) $x2list,
>      wmatrix(robust)
. estat overid
Test of overidentifying restriction:
Hansen's J chi2(1) = 1.65198 (p = 0.1987)
```

The test statistic is  $\chi^2(1)$  distributed because the number of overidentifying restrictions equals  $2 - 1 = 1$ . Because  $p > 0.05$ , we do not reject the null hypothesis and conclude that the overidentifying restriction is valid.

A similar test using all four available instruments yields

```
. * Test of overidentifying restrictions following ivregress gmm
. ivregress gmm ldrugexp (hi_empunion = ssiratio lowincome multlc firmsz)
>      $x2list, wmatrix(robust)

Instrumental variables GMM regression                               Number of obs     =    10,068
                                                               Wald chi2(6)    =   2053.18
                                                               Prob > chi2    =     0.0000
                                                               R-squared       =     0.0894
                                                               Root MSE        =     1.2992

GMM weight matrix: Robust
```

ldrugexp	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
hi_empunion	-.7809169	.1690563	-4.62	0.000	-1.112261	-.4495727
totchr	.4488934	.0099905	44.93	0.000	.4293123	.4684745
age	-.0120102	.0026498	-4.53	0.000	-.0172037	-.0068166
female	-.0087044	.0300802	-0.29	0.772	-.0676604	.0502516
blhisp	-.2014736	.037838	-5.32	0.000	-.2756347	-.1273125
linc	.0783428	.0195822	4.00	0.000	.0399624	.1167232
_cons	6.669519	.2320541	28.74	0.000	6.214701	7.124336

```
Instrumented: hi_empunion
Instruments: totchr age female blhisp linc ssiratio lowincome multlc firmsz
. estat overid
Test of overidentifying restriction:
Hansen's J chi2(3) = 11.0269 (p = 0.0116)
```

Now we reject the null hypothesis at level 0.05, but we do not reject at level 0.01, though the decision is marginal. Despite this rejection, the coefficient of the endogenous regressor `hi_empunion` is  $-0.781$ , not all that different from the estimate when `ssiratio` is the only instrument.

#### 7.4.9 IV estimation using the eregress command

The `eregress` command, detailed in section 23.7, is a command for extended regression that fits a linear regression while controlling for one or more of the following complications: endogenous regressor, endogenous sample selection, and nonrandom treatment assignment.

The command provides maximum likelihood (ML) estimates under the assumption of normally distributed errors. With an endogenous regressor, the `eregress` command yields the LIML estimator presented in section [7.9.1](#). In the just-identified case, the LIML estimator and 2SLS estimators are equivalent because they reduce to the IV estimator.

The following command yields the same IV results for the example of section [7.4.4](#) as those obtained using the `ivregress` command because the model is just identified.

```
. * IV estimation using the egress command in a just-identified model
. egress ldrugexp $x2list, endogenous(hi_empunion = $x2list ssiratio) vce(robust)
(output omitted)
```

#### 7.4.10 IV estimation with a binary endogenous regressor

In our example, the endogenous regressor `hi_empunion` is a binary variable. The IV methods we have used are valid under the assumption that  $E(u_i|\mathbf{z}_i) = 0$ , which in our example means that the error in the structural equation for `ldrugexp` has a mean of zero conditional on the exogenous regressors  $\mathbf{x}_1$  in [\(7.5\)](#) and on any instruments  $\mathbf{x}_2$  such as `ssiratio`. The reasonableness of this assumption does not change when the endogenous regressor `hi_empunion` is binary.

An alternative approach to linear IV adds more structure to explicitly account for the binary nature of the endogenous regressor by changing the first-stage model to be a latent-variable model similar to the probit model presented in section 17.2. Let  $y_1$  depend in part on  $y_2$ , a binary endogenous regressor. We introduce an unobserved latent variable,  $y_2^*$ , that determines whether  $y_2 = 1$  or 0. The models [\(7.5\)](#) and [\(7.6\)](#) become

$$\begin{aligned} y_{1i} &= \beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + u_i \\ y_{2i}^* &= \mathbf{x}'_{1i} \boldsymbol{\pi}_{1j} + \mathbf{x}'_{2i} \boldsymbol{\pi}_{2j} + v_i \\ y_{2i} &= \begin{cases} 1 & \text{if } y_{2i}^* > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7.12}$$

The errors  $(u_i, v_i)$  are assumed to be correlated bivariate normal with  $\text{Var}(u_i) = \sigma^2$ ,  $\text{Var}(v_i) = 1$ , the standard normalization for a probit model,

and  $\text{Cov}(u_i, v_i) = \rho\sigma^2$ .

The binary endogenous regressor  $y_2$  can be viewed as a treatment indicator. If  $y_2 = 1$ , we receive treatment (here access to employer- or union-provided insurance), and if  $y_2 = 0$ , we do not receive treatment. The Stata documentation refers to (7.12) as the treatment-effects model, though the treatment-effects literature is vast and encompasses many models and methods.

The `etregress` command fits (7.12) by ML, the default, or by two-step methods. The basic syntax is

```
etregress depvar [indepvars], treat(depvar_t = indepvars_t) [twostep|cfunction]
```

where  $depvar$  is  $y_1$ ,  $indepvars$  is  $\mathbf{x}_1$ ,  $depvar_t$  is  $y_2^*$ , and  $indepvars_t$  is  $\mathbf{x}_1$  and  $\mathbf{x}_2$ .

We apply this estimator to the exactly identified setup of section 7.4.4, with the single instrument `ssiratio`. We obtain

```

. * Regression with a binary endogenous regressor
. etregress ldrugexp $x2list, treat(hi_empunion = ssiratio $x2list) vce(robust)
Iteration 0: log pseudolikelihood = -22666.718
Iteration 1: log pseudolikelihood = -22655.915
Iteration 2: log pseudolikelihood = -22654.399
Iteration 3: log pseudolikelihood = -22654.396
Iteration 4: log pseudolikelihood = -22654.396

Linear regression with endogenous treatment                               Number of obs = 10,068
Estimator: Maximum likelihood                                         Wald chi2(6) = 1876.40
Log pseudolikelihood = -22654.396                                         Prob > chi2 = 0.0000

```

		Robust			
	Coefficient	std. err.	z	P> z	[95% conf. interval]
ldrugexp					
totchr	.4550432	.0108265	42.03	0.000	.4338237 .4762627
age	-.0179835	.0024374	-7.38	0.000	-.0227606 -.0132064
female	-.06001	.0303896	-1.97	0.048	-.1195725 -.0004475
blhisp	-.2479348	.039701	-6.25	0.000	-.3257474 -.1701222
linc	.1267941	.0186249	6.81	0.000	.09029 .1632981
1.hi_empunion	-1.384199	.1029986	-13.44	0.000	-1.586072 -1.182325
_cons	7.240257	.2033296	35.61	0.000	6.841738 7.638776
hi_empunion					
ssiratio	-.5506714	.0400378	-13.75	0.000	-.629144 -.4721987
totchr	.0405104	.0100773	4.02	0.000	.0207593 .0602616
age	-.0242543	.0020402	-11.89	0.000	-.0282531 -.0202556
female	-.1924384	.0262529	-7.33	0.000	-.2438931 -.1409837
blhisp	-.1928805	.0358713	-5.38	0.000	-.2631871 -.122574
linc	.1278699	.0161179	7.93	0.000	.0962794 .1594604
_cons	1.508993	.1649894	9.15	0.000	1.18562 1.832367
/athrho	.7639572	.0612615	12.47	0.000	.6438869 .8840276
/lnsigma	.3460358	.0197529	17.52	0.000	.3073209 .3847507
rho	.6434019	.0359013			.5675403 .7084313
sigma	1.413453	.0279197			1.359777 1.469248
lambda	.9094185	.0672652			.7775812 1.041256

Wald test of indep. eqns. (rho = 0): chi2(1) = 155.51 Prob > chi2 = 0.0000

The key output is the first set of regression coefficients. Compared with IV estimates in section [7.4.4](#), the coefficient of hi\_empunion has increased in absolute value from – 0.812 to – 1.384, and the standard error has fallen greatly from 0.188 to 0.103. The coefficients and standard errors of the exogenous regressors change much less.

The quantities `rho`, `sigma`, and `lambda` denote, respectively,  $\rho$ ,  $\sigma$ , and  $\rho\sigma$ . To ensure that  $\hat{\sigma} > 0$  and  $|\hat{\rho}| < 1$ , `etregress` estimates the transformed parameters  $0.5 \times \ln\{(1 + \rho)/(1 - \rho)\}$ , reported as `/athrho`, and  $\ln\sigma$ , reported as `/lnsigma`. If the error correlation  $\rho = 0$ , then the errors  $u$  and  $v$  are independent, and there is no endogeneity problem. The last line of output clearly rejects  $H_0: \rho = 0$ , so `hi_empunion` is indeed an endogenous regressor.

Further analysis of this example, one in a treatment-effects framework, is provided in section 25.4.2.

Which method is better: regular IV or the latent variable model (7.12)? Intuitively, (7.12) imposes more structure. The benefit may be increased precision of estimation, as in this example. The cost is a greater chance of misspecification error. If the errors are heteroskedastic, as is likely, the IV estimator remains consistent, but the treatment-effects estimator given here becomes inconsistent.

More generally, when regressors in nonlinear models, such as binary-data models and count-data models, include endogenous regressors, there is more than one approach to model estimation; see also sections 17.9, 20.7, and 23.7.

#### 7.4.11 IV as local average treatment-effects estimator

When both the endogenous regressor  $y_2$  and its single instrument  $x_2$  are binary, the IV estimator can be reinterpreted as a local average treatment-effects (LATE) estimator that allows the effect of the endogenous regressor to vary across individuals. Then the IV estimator gives the average effect of the binary endogenous regressor, where the average is across a subgroup of the population called compliers and it is assumed that there are no defiers. Compliers and defiers constitute notional, not observable, categories; they are required to identify the treatment effect, that is, to interpret which group the treatment effect refers to.

For example, suppose we are interested in the effect of attending a charter school on an outcome  $y_1$  such as exam performance. Let  $y_2 = 1$  if a student attends a charter school and  $y_2 = 0$  otherwise, and let the instrument

$x_2 = 1$  if the student wins a lottery to attend a charter school and  $x_2 = 0$  otherwise. Then a complier is someone who attends the charter school only if he or she wins the lottery; that is,  $y_2 = 1$  if  $x_2 = 1$  and  $y_2 = 0$  if  $x_2 = 0$ . And a defier is someone who attends the charter school only if he or she loses the lottery; that is,  $y_2 = 1$  if  $x_2 = 0$  and  $y_2 = 0$  if  $x_2 = 1$ .

The formal definition of compliers and defiers, and this very important interpretation and use of IV in the treatment evaluation literature, is deferred to section 25.5. It can explain why different binary instruments may lead to different IV estimates, even asymptotically; hence the use of the adjective “local”.

## 7.5 Weak instruments

In this section and the subsequent two sections, we consider inference for IV estimation when the correlation between instruments and endogenous regressors is weak enough that standard asymptotic theory may provide a poor guide to actual finite-sample distributions, even in samples that have many observations and assuming that the chosen instruments are valid, so IV estimators remain consistent. The discussion is lengthy. For the common case of a just-identified model with a single endogenous regressor it is standard to use the Anderson–Rubin Wald test given in section [7.7](#)

This section provides a brief overview of the essential issues, a simulation exercise, and discusses finite-sample bias. Several weak-instrument diagnostics and tests are presented in section [7.6](#), most notably use of the first-stage  $F$  statistic. Inferential methods when instruments are weak are presented in section [7.7](#), most notably Anderson–Rubin Wald tests.

Any study using IV methods should recognize the potential of weak instruments. The literature is vast; [Andrews, Stock, and Sun \(2019\)](#) provide a recent survey. A major complication is that the first-order asymptotic equivalence of estimators such as 2SLS and LIML and of Wald, Lagrange multiplier (LM), and likelihood-ratio (LR) tests disappears when instruments are weak. We focus on IV (or 2SLS) estimators and Wald tests because these are the most commonly used methods in practice.

### 7.5.1 Weak instruments essentials

When instruments are potentially weak, there are two general approaches. Empirical studies to date have used a two-step approach that first determines whether instruments are weak and then proceeds to use standard asymptotic theory if instruments are determined not to be weak. A better theoretically grounded approach, one developed more recently, bases inference on an alternative asymptotic theory that is valid regardless of whether instruments are weak.

For simplicity, we focus on the most common case of a single endogenous regressor. Then, using the model setup of section [7.3.2](#), we have structural equation  $y_1 = \beta_1 y_2 + \mathbf{x}'_1 \boldsymbol{\beta}_2 + u$  and first-stage equation  $y_2 = \mathbf{x}'_1 \boldsymbol{\pi}_1 + \mathbf{x}'_2 \boldsymbol{\pi}_2 + v$ .

To see the weak instrument problem, note that if the only regressor is the endogenous variable  $y_2$  and there is only one instrument  $x_2$ , then  $\widehat{\beta}_{1,IV}$  is the sample analog of  $\text{Cov}(x_2, y_1)/\text{Cov}(x_2, y_2)$  with problematic distribution when the denominator in the ratio, the covariance between the instrument and the endogenous regressor, is too close to zero.

The two-stage method first tests the strength of the instruments  $\mathbf{x}_2$  in the first-stage model by computing the  $F$  statistic of the hypothesis that  $\boldsymbol{\pi}_2 = \mathbf{0}$ . If  $F$  exceeds a threshold  $\overline{F}$ , then the instrument is viewed as strong enough that we can perform regular inference on the IV estimate of  $\beta_1$  in the structural equation. An initial popular choice of the threshold was  $\overline{F} = 10$ , which corresponds in the just-identified case to deciding that an instrument is not weak if its  $t$  statistic in the first-stage regression exceeds  $\sqrt{10} = 3.16$ . This threshold of 10 is now felt to be too low to conclude that there may be no weak instrument problem. Section [7.6.4](#) presents more formal tests based on the  $F$  statistic.

The problem with this approach is that a nominal 5% test at the second stage has true size substantially larger than 5% because of the first-stage  $F$  test; see sections [7.6.4](#) and [7.7.1](#).

The alternative better approach uses asymptotic theory that is valid regardless of whether instruments are weak. A standard method is the Anderson–Rubin test. Substituting the first-stage equation into the structural equation yields the reduced-form equation for  $y_1$  as  $y_1 = \mathbf{x}'_1(\beta_1 \boldsymbol{\pi}_1 + \boldsymbol{\beta}_2) + \mathbf{x}'_2 \beta_1 \boldsymbol{\pi}_2 + (u + \beta_1 v)$ . Standard inference methods can be applied to OLS estimates of this equation because the estimates do not involve  $y_2$ . The Anderson–Rubin test of whether  $\beta_1 = 0$  is a standard Wald  $F$  test of the joint statistical significance of the instruments  $\mathbf{x}_2$ ; see section [7.7.2](#).

The Anderson–Rubin approach can be applied using robust standard errors and is optimal in the case of a just-identified model with a single

endogenous regressor. It has low power in overidentified models, however, leading to alternative methods detailed in section [7.7](#) that are still being developed.

### 7.5.2 Finite-sample properties of IV estimators

Even when IV estimators are consistent, they are generally biased in finite samples. This result has been formally established in overidentified models. In just-identified models, the first moment of the IV estimator does not exist, but for simplicity, we follow the literature and continue to use the term “bias” in this case.

The finite-sample properties of IV estimators are complicated. However, there are three cases in which it is possible to say something about the finite-sample bias of IV; see [Davidson and MacKinnon \(2004, chap. 8.4\)](#).

First, when the number of instruments is very large relative to the sample size and the first-stage regression fits very well, the IV estimator may approach the OLS estimator and hence will be similarly biased. This case of many instruments is not very relevant for cross-sectional microeconometrics data, where usually few instruments are available, though it can be relevant for panel-data IV estimators such as Arellano–Bond.

Second, when the correlation between the structural-equation error  $u$  in [\(7.5\)](#) and some components of the vector  $v$  in [\(7.6\)](#) of first-stage-equation errors is high, then asymptotic theory may be a poor guide to the finite-sample distribution.

Third, if we have weak instruments in the sense that one or more of the first-stage regressions have a poor fit, with  $\pi_{2j} \simeq 0$  in [\(7.6\)](#), then asymptotic theory may also provide a poor guide to the finite-sample distribution of the IV estimator, especially when the sample is small but possibly even if the sample has thousands of observations.

In what follows, our main focus will be on the third source of finite-sample bias of IV, that due to weak instruments. More precise definitions of weak instruments are considered later in this section.

### 7.5.3 A Monte Carlo example

To expand on the results stated in the preceding subsection, we now consider a simple Monte Carlo example with equations for two endogenous variables denoted  $y_1$  and  $y_2$ , respectively, and an exogenous instrumental variable for  $y_2$ , denoted  $z$ . Specifically, we have

$$\begin{aligned}y_1 &= \beta y_2 + u \\y_2 &= \pi z + v\end{aligned}$$

where  $(u, v)$  are random shocks. The parameter  $\pi$  measures the strength of the linear association between  $y_2$  and  $z$ ; small values of  $\pi$  imply that  $z$  is a weak instrument for  $y_2$ . The (“causal”) parameter  $\beta$  is the key object of inference.

The reduced-form equation for  $y_1$  is

$$\begin{aligned}y_1 &= \beta(\pi z + v) + u \\&= \beta\pi z + (\beta v + u)\end{aligned}$$

A key determinant of weak instrument bias is the concentration parameter  $\tau^2$ , with general definition given below in (7.15). For the current simulation setup,

$$\tau^2 = N\pi \times E(z^2) \times \pi = N\pi^2 E(z^2)$$

Weak instrument bias increases as  $\tau^2$  decreases. The  $F$  statistic from first-stage regression of  $y_2$  on  $z$  is asymptotically equivalent to  $\tau^2 + 1$ .

We consider two cases. The first is a recursive model that results from the noncorrelation assumption that  $E(uv) = 0$ . In this case, we may treat  $y_2$  as exogenous when estimating the first equation, and  $\beta$  can be consistently estimated by OLS. Then the IV estimator is defined as  $\widehat{\beta}_{IV} = \sum_i zy_1 / \sum_i zy_2$ , which is a consistent estimator but is not efficient relative to  $\widehat{\beta}_{OLS}$ .

In the second case, we assume that the random shocks  $(u, v)$  are contemporaneously correlated. In this case,  $E(y_2 u) \neq 0$ ; hence,  $\widehat{\beta}_{OLS}$  is inconsistent (asymptotically biased), and  $\widehat{\beta}_{IV}$  is consistent (asymptotically unbiased). However, in small samples,  $\widehat{\beta}_{IV}$  may also be biased.

To throw additional light on the source and nature of the bias, we carry out a simple simulation experiment. The data are generated under the following assumptions:  $\beta = 2$ , the sample size  $N$  equals 100 (small sample) or 1,000 (large sample), and  $\pi$  equals 0.1 (“weak” instrument) or equals 0.5 (“strong” instrument). In the first set of simulations,  $u$  and  $v$  are independent standard normals, and in the second set,  $(u, v)$  are drawn from a standard bivariate normal distribution with correlation of 0.5. The number of replications in all simulations is set at 1,000.

The simulation program, set up to provide summary statistics for the parameter  $\beta$ , is given below. Its structure is similar to the program in section [5.6.1](#). The program can be easily changed to simulate alternative settings of  $\beta$ ,  $\pi$ ,  $\text{Cor}(u, v)$ , and  $N$ .

```

. * Program for weak instruments simulation
. clear all
. global numssims 1000
. program weakivsim, rclass
  1.   version 17
  2.   drop _all
  3.   set obs $numobs                      // Set sample size
  4.   generate z = rnormal()
  5.   generate v = rnormal()
  6.   generate u = rnormal() + $ecor*v      // corr(u,v)=ecor/sqrt(1+ecor^2)
  7.   generate y2 = 0 + $pi*z + v          // Set instrument strength
  8.   generate y1 = 0 + 2*$pi*z + (u+2*v) // beta = 2
  9.   regress y1 y2
10.   return scalar bols = _b[y2]
11.   return scalar seols = _se[y2]
12.   return scalar tols = (_b[y2]-2)/_se[y2]
13.   ivregress 2sls y1 (y2=z)
14.   return scalar biv = _b[y2]
15.   return scalar seiv = _se[y2]
16.   return scalar tiv = (_b[y2]-2)/_se[y2]
17.   regress y2 z                         // First-stage regression
18.   return scalar Fiv = e(F)
19. end

```

### 7.5.4 Simulation results with uncorrelated errors

We begin with a large sample. Then both OLS and IV are consistent, and OLS is fully efficient given this simulation design.

```
. * Simulation 1: beta=2, large sample, weak IV, independent errors
. global numobs 1000 // Large sample
. global pi 0.1 // Weak IV
. global ecor 0.0 // correlation(u,v)=0
. simulate bols=r(bols) seols=r(seols) tols=r(tols) biv=r(biv) seiv=r(seiv)
>      tiv=r(tiv) Fiv=r(Fiv), seed(10101) reps($numobs) nolegend nodots:
>      weakivsim

. mean bols seols biv seiv Fiv
```

Mean estimation Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
bols	2.000625	.0010369	1.998591 2.00266
seols	.0314616	.0000309	.031401 .0315222
biv	1.989567	.0165767	1.957038 2.022096
seiv	.5107373	.061441	.3901691 .6313055
Fiv	11.06388	.2105104	10.65079 11.47698

```
. display "Concentration parameter = " $numobs*$pi*1*$pi // As E[z^2]=1
Concentration parameter = 10
```

Both estimators appear to be unbiased, with the average value close to 2.0 across the simulations. The IV estimator is much less efficient, with an average standard error that is 16 times ( $= 0.511/0.031$ ) that of OLS. So a weak instrument has greatly reduced efficiency but has not induced finite-sample bias problems with  $N = 1000$ .

Next, we move to a smaller sample, reducing  $N$  from 1,000 to 100.

```

. * Simulation 2: beta=2, small sample, weak IV, independent errors
. global numobs 100      // Small sample
. global pi 0.1          // Weak IV
. global ecor 0.0         // correlation(u,v)=0
. simulate bols=r(bols) seols=r(seols) tols=r(tols) biv=r(biv) seiv=r(seiv)
>      tiv=r(tiv) Fiv=r(Fiv), seed(10101) reps($numobs) nolegend nodots:
>      weakivsim

. mean bols seols biv seiv Fiv

```

Mean estimation Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
bols	2.000745	.00311	1.994642 2.006847
seols	.1010183	.0003322	.1003665 .1016701
biv	2.374817	.3618055	1.664832 3.084803
seiv	137.1123	43.71515	51.32823 222.8963
Fiv	2.132318	.0862476	1.963071 2.301565

```

. display "Concentration parameter = " $numobs*$pi*1*$pi // As E[z^2]=1
Concentration parameter = 1

```

The OLS estimator remains unbiased though, as expected, the average standard error is  $\sqrt{10}$  times larger because the sample is one-tenth as large. The IV estimator is now an extremely noisy estimator with average standard error of 137 and very wide 95% simulation interval of [1.66, 3.08], though this does include the DGP value of 2.

We repeat this small-sample simulation with  $\pi$  increased from 0.1 to 0.5. We obtain

```

. * Simulation 3: beta=2, small sample, strong IV, independent errors
. global numobs 100 // Small sample
. global pi 0.5 // Strong IV
. global ecor 0.0 // correlation(u,v)=00
. simulate bols=r(bols) seols=r(seols) tols=r(tols) biv=r(biv) seiv=r(seiv)
>      tiv=r(tiv) Fiv=r(Fiv), seed(10101) reps($numobs) nolegend nodots: weakivsim
. mean bols seols biv seiv Fiv

```

Mean estimation Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
bols	2.000864	.0028334	1.995304 2.006424
seols	.0906651	.0002983	.0900798 .0912503
biv	2.000488	.0073413	1.986082 2.014894
seiv	.2151584	.0020243	.2111861 .2191308
Fiv	26.79966	.3818293	26.05038 27.54894

```

. display "Concentration parameter = " $numobs*$pi*1*$pi // As E[z^2]=1
Concentration parameter = 25

```

Now the IV estimator is well behaved with average closely centered on 2.0 and a small average standard error that is only 2.4 times ( $= 0.215/0.091$ ) that of OLS, so the efficiency loss is not as great.

### 7.5.5 Simulation results with correlated errors

We next present simulations where the OLS estimator is inconsistent because of correlation of the errors, leading to  $y_2$  becoming an endogenous regressor in the  $y_1$  equation. The IV estimator remains consistent but is no longer centered on 2 in small samples.

We first consider a large sample, with weak instrument and error correlation of 0.5. We obtain

```

. * Simulation 4: beta=2, large sample, weak IV, correlated errors
. global numobs 1000 // Large sample
. global pi 0.1 // Weak IV
. global ecor 0.5775 // Implies correlation(u,v)=.5775/sqrt(1+.5775^2)=0.5
. simulate bols=r(bols) seols=r(seols) tols=r(tols) biv=r(biv) seiv=r(seiv)
>      tiv=r(tiv) Fiv=r(Fiv), seed(10101) reps($numobs) nolegend nodots:
>      weakivsim

. mean bols seols biv seiv Fiv

```

Mean estimation

Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
bols	2.572409	.0010378	2.570372 2.574445
seols	.0315148	.0000309	.0314541 .0315755
biv	1.869067	.0299957	1.810205 1.927929
seiv	.7937734	.2411204	.3206129 1.266934
Fiv	11.06388	.2105104	10.65079 11.47698

```

. display "Concentration parameter = " $numobs*$pi*1*$pi // As E[z^2]=1
Concentration parameter = 10

```

As expected, the OLS estimator is inconsistent. It is centered a long way from 2.0, with average value of 2.572 and average standard error of 0.032. (This simulation shows only bias, but if we make the sample size very large, the OLS estimates are still a long way from 2.0.)

The IV estimator is closer to being centered on 2.0, but the 95% simulation interval of [1.810, 1.928] does not include 2.0. With both appreciable endogeneity and a weak instrument, the IV estimator displays bias even with  $N = 1000$ .

Finally, we verify that IV estimation works well with appreciable endogeneity if the instrument is strong. We increase  $\pi$  from 0.1 to 0.5.

```

. * Simulation 5: beta=2, small sample, strong IV, correlated errors
. global numobs 100 // Small sample
. global pi 0.5 // Strong IV
. global ecor 0.5775 // Implies correlation(u,v)=.5775/sqrt(1+.5775^2)=0.5
. simulate bols=r(bols) seols=r(seols) tols=r(tols) biv=r(biv) seiv=r(seiv)
>      tiv=r(tiv) Fiv=r(Fiv), seed(10101) reps($numobs) nolegend nodots:
>      weakivsim

. mean bols seols biv seiv Fiv

```

Mean estimation

Number of obs = 1,000

	Mean	Std. err.	[95% conf. interval]
bols	2.461669	.0029383	2.455903 2.467435
seols	.0936148	.0003095	.0930074 .0942222
biv	1.974495	.0088162	1.957195 1.991796
seiv	.2533962	.0033324	.2468569 .2599356
Fiv	26.79966	.3818293	26.05038 27.54894

```

. display "Concentration parameter = " $numobs*$pi*1*$pi // As E[z^2]=1
Concentration parameter = 25

```

The OLS estimator is again clearly inconsistent. The IV estimator is close to being centered on 2.0 with 95% simulation interval of [1.957, 1.992].

These simulation results suggest that with weak instruments, the IV estimator is biased, which will in turn lead to tests of incorrect size and confidence intervals with incorrect coverage. The bias increases as the concentration parameter decreases.

For a single endogenous regressor, the concentration parameter defined below in (7.15) is asymptotically equal to  $F - 1$ , where  $F$  is the first-stage  $F$  statistic. In simulations 1–5, the design sets the concentration parameter to, respectively, 10, 1, 25, 10 and 25, and the simulation average of the corresponding  $F$  statistics were, respectively, 11.06, 2.13, 26.80, 11.06, and 26.80.

### 7.5.6 The first-stage F statistic

The most obvious diagnostic for whether an instrument is weak is to test whether it is weakly correlated with the endogenous regressor.

For IV estimation that uses more than one instrument, we can consider the joint correlation of the endogenous regressor with the several instruments. Possible measures of this correlation are  $R^2$  from regression of the endogenous regressor  $y_2$  on the several instruments  $\mathbf{x}_2$ , and the  $F$  statistic for test of overall fit in this regression. Low values of  $R^2$  or  $F$  are indicative of weak instruments.

This analysis neglects the presence of the structural-model exogenous regressors  $\mathbf{x}_1$  in the first-stage regression (7.6) of  $y_2$  on both  $\mathbf{x}_2$  and  $\mathbf{x}_1$ . If the instruments  $\mathbf{x}_2$  add little extra to explaining  $y_1$  after controlling for  $\mathbf{x}_1$ , then the instruments are weak.

One commonly used diagnostic is therefore the  $F$  statistic for joint significance of the instruments  $\mathbf{x}_2$  in first-stage regression of the endogenous regressor  $y_2$  on  $\mathbf{x}_2$  and  $\mathbf{x}_1$ . This is a test that  $\boldsymbol{\pi}_2 = \mathbf{0}$  in (7.6). When we collect all observations, the first-stage model is  $\mathbf{y}_2 = \mathbf{X}_1\boldsymbol{\pi}_1 + \mathbf{X}_2\boldsymbol{\pi}_2 + \mathbf{v}_2$ . After partialing out the effect of  $\mathbf{x}_1$  on  $y_2$  and of  $\mathbf{x}_1$  on  $\mathbf{x}_2$ , we reduce the first-stage model to

$$\tilde{\mathbf{y}}_2 = \tilde{\mathbf{X}}_2\boldsymbol{\pi}_2 + \tilde{\mathbf{v}}_2 \quad (7.13)$$

where  $\tilde{y}_{2i}$  are residuals from OLS of  $y_{2i}$  on  $\mathbf{x}_{1i}$  and  $\tilde{\mathbf{x}}_{2i}$  are the residuals from OLS of  $\mathbf{x}_{2i}$  on  $\mathbf{x}_{1i}$ .

Then with i.i.d. errors, a test of  $\boldsymbol{\pi}_2 = \mathbf{0}$  when  $y_2$  is scalar is based on

$$F = \left( \tilde{\boldsymbol{\pi}}_2' \tilde{\mathbf{X}}_2' \tilde{\mathbf{X}}_2 \tilde{\boldsymbol{\pi}}_2 / K_2 \right) / s_{\tilde{v}_2}^2 \quad (7.14)$$

This is called the first-stage  $F$  statistic. Section 7.6.4 presents tests of whether  $F$  is low enough to indicate that instruments are likely to be weak. Section 7.7.1 discusses the limitations of using conventional inference after first screening on the  $F$  statistic.

### 7.5.7 The first-stage F statistic and 2SLS bias

The concentration parameter

$$\tau^2 = \left( \boldsymbol{\pi}'_2 \tilde{\mathbf{X}}_2 \tilde{\mathbf{X}}'_2 \boldsymbol{\pi}_2 \right) / \sigma_{\tilde{v}^2}^2 \quad (7.15)$$

is the population analog of the first-stage  $F$  statistic given in (7.14), aside from division by the number of instruments ( $K_2$ ). It can be shown that  $F - 1$  is an approximately unbiased estimate of  $\tau^2/K_2$ .

Theory shows that finite-sample bias of the IV estimator increases as  $\tau^2$  decreases and that  $\tau$  plays the role of sample size.

To see this, consider the model  $y_1 = \beta y_2 + u$ , with  $K_2$  instruments so  $y_2 = \mathbf{z}' \boldsymbol{\pi} + v$ . Then the bias of the 2SLS estimator is approximately

$$E(\hat{\beta}_{2SLS}) - \beta \simeq \text{Cor}(u, v) \times \frac{\sigma_u}{\sigma_v} \times \frac{1}{(\tau^2/K_2) + 1} \quad (7.16)$$

See [Angrist and Pischke \(2009\)](#), 207), where strictly speaking  $E(\hat{\beta}_{2SLS})$  exists only if  $K_2 \geq 2$ .

There are several key results. First, the bias of 2SLS increases with the correlation of the two errors, but this is why we needed to do 2SLS in the first place. Second, the bias of 2SLS is likely to get larger as more instruments are added (unless they substantially increase the  $F$  statistic). Third, the bias of 2SLS is greater the smaller the sample size  $N$  because the concentration parameter depends on  $\tilde{\mathbf{X}}_2 \tilde{\mathbf{X}}'_2$ , which is the sum of  $N$  terms. The preceding simulations varied  $N$  and  $\text{Cov}(u, v)$ .

Result (7.16) shows the importance of the concentration parameter  $\tau^2$ , which grows with sample size  $N$  at rate  $N$ . The weak instruments asymptotics considered later consider the limit distribution of  $\sqrt{\tau}(\hat{\beta}_{2SLS} - \beta)$ , where  $\tau = c/\sqrt{N}$  for  $c$  a constant, rather than the usual  $\sqrt{N}(\hat{\beta}_{2SLS} - \beta)$ , because then  $\tau^2$  does not grow with the sample size.

As already noted, it can be shown that  $F - 1$  is an approximately unbiased estimate of  $\tau^2/K_2$ . So (7.16) implies

$$\text{2SLS bias} \simeq \text{Cor}(u, v) \times \frac{\sigma_u}{\sigma_v} \times \frac{1}{F}$$

This shows the importance of the first-stage  $F$  statistic in signaling likely bias in the 2SLS estimator.

Finally, we note that throughout we have assumed that instruments are uncorrelated with the structural equation error, so  $\text{Cor}(\mathbf{x}_2, u) = \mathbf{0}$  in (7.5)–(7.6) because this is necessary for the 2SLS estimator to be consistent. If instead there is even mild correlation between  $\mathbf{x}_2$  and  $u$ , and additionally the instruments are weak, so  $\text{Cor}(\mathbf{x}_2, y_1)$  is low, then the 2SLS estimator can be even more inconsistent than the OLS estimator; see [Bound, Jaeger, and Baker \(1995\)](#) or [Cameron and Trivedi \(2005, 106–107\)](#). This increased fragility of the assumption that the instrument is valid when instruments are weak is rarely emphasized.

### 7.5.8 Test size distortion

When instruments are weak, standard asymptotic tests have size distortion because of the aforementioned bias of the 2SLS estimator and also because the asymptotic normal distribution provides a poor approximation to the distribution of the 2SLS estimator.

So the standard tests on the coefficients of endogenous regressors in the structural model have the wrong size, and the associated confidence intervals have the wrong coverage probabilities. Section 7.7 presents valid inference methods for the coefficients when instruments are weak. Additionally, the usual implementations of Hausman tests for endogeneity and overidentification tests will also have the wrong size if instruments are weak.

## 7.6 Diagnostics and tests for weak instruments

There are several approaches for assessing whether there is a weak-instrument problem, based on analysis of the first-stage reduced-form equations and, particularly, the  $F$  statistic for the joint significance of the key instruments.

Initial work, and the `ivregress` postestimation command `estat firststage`, presented in section [7.6.5](#), relies on the assumption of i.i.d. errors. Relaxing this assumption is the subject of ongoing research, and some of the methods presented here may be supplanted.

Note that this section focuses solely on detecting whether instruments are weak. The subsequent section presents valid inference methods with potentially weak instruments.

In particular, a 5% Wald test of structural model parameters has true size considerably greater than 5% if it is preceded by a screening test for weak instruments. If instruments are thought to be potentially weak, then current research is advising to skip screening and move directly to the inference methods presented in section [7.7](#).

### 7.6.1 Pairwise correlations

The simplest method is to use pairwise correlations between any endogenous regressor and instruments. For our example, we have

```

. * Correlations of endogenous regressor with instruments
. qui use mus207mepspresdrugs, clear
. correlate hi_empunion ssiratio lowincome multlc firmsz if linc!=.
(obs=10,068)

```

	hi_emp~n	ssiratio	lowinc~e	multlc	firmsz
hi_empunion	1.0000				
ssiratio	-0.2243	1.0000			
lowincome	-0.1164	0.2676	1.0000		
multlc	0.1199	-0.1982	-0.0625	1.0000	
firmsz	0.0374	-0.0464	-0.0082	0.1873	1.0000

```

. display "Concentration parameter = " $numobs*2*$pi/(2^2+2*2*$ecor+1)
Concentration parameter = 13.679891

```

The gross correlations of instruments with the endogenous regressor `hi_empunion` are low. This will lead to considerable efficiency loss using IV compared with OLS. But the correlations, aside perhaps from 0.037 for `firmsz`, are not so low as to immediately flag a problem of weak instruments.

## 7.6.2 Partial $R^2$

We are interested in the correlation between the endogenous regressor and the instruments after controlling for the presence of other regressors.

One diagnostic is the partial  $R^2$  between  $y_2$  and  $\mathbf{x}_2$  after controlling for  $\mathbf{x}_1$ . This is the  $R^2$  from OLS regression ([7.13](#)). For structural equations with more than one endogenous regressor and hence more than one first-stage regression, a generalization called Shea's partial  $R^2$  has been proposed. There is no consensus on how low of a value indicates a problem, however, so this diagnostic is less often used.

## 7.6.3 Tests of underidentification

Identification requires that the instruments be correlated with the endogenous regressors.

In the simplest case of a single endogenous variable, this can be tested using the first-stage  $F$  statistic defined in ([7.14](#)). The [Cragg and](#)

[Donald \(1993\)](#) minimum eigenvalue statistic is a generalization to multiple endogenous variables, defined in [Stock and Yogo \(2005](#), 84) or in [R] **ivregress postestimation**, that reduces to the first-stage  $F$  statistic when  $m = 1$ .

[Kleibergen and Paap \(2006\)](#) proposed an alternative test statistic that reduces to the Cragg–Donald test when errors are i.i.d. but can be applied to models with robust standard errors (and to nonlinear GMM).

The problem with these tests is that they are too weak. In the simplest case of a single endogenous regressor, single instrument, and  $N$  large, they amount to rejecting the null hypothesis of underidentification at level 0.05 if the first-stage  $F = 3.84$  or, equivalently, the single instrument has  $t = 1.96$ .

#### 7.6.4 Tests of weak instruments

Small values of the first-stage  $F$  statistic indicate weak instruments. There is no clear critical value for the  $F$  statistic defined in (7.14), because it depends on the criteria used to determine whether instruments are weak, on the number of endogenous variables, and on the number of overidentifying restrictions (excess instruments).

[Stock and Yogo \(2005\)](#) proposed two approaches for testing whether instruments are weak based on the size of the first-stage  $F$  statistic when errors are i.i.d. Few results are available in the more common case of heteroskedastic or clustered errors; a notable exception is the test of [Montiel Olea and Pflueger \(2013\)](#) given later in this subsection. Test rejection is interpreted as meaning there is a weak instrument problem.

We focus on the 2SLS estimator. The tests have been adapted to other estimators that control for endogenous regressors, most notably the LIML estimator presented in section [7.9.1](#).

##### Relative bias of 2SLS to OLS test with i.i.d. errors

The first approach considers the relative bias of 2SLS to OLS estimators of the endogenous regressor coefficient(s)  $\beta_1$ , where 2SLS has finite-sample bias

that disappears asymptotically while OLS is always biased. For a single endogenous regressor, the relative bias is

$$B^2 = \frac{\left\{E(\widehat{\beta}_{1,2SLS}) - \beta_1\right\}^2}{\left\{E(\widehat{\beta}_{1,OLS}) - \beta_1\right\}^2}$$

We choose a tolerable level of relative bias, say, 10% ( $B = 0.10$ ), and test at the 5% significance level the null hypothesis that  $B \leq 0.10$ . The critical values of  $F$  vary with  $B$ , the number of endogenous regressors ( $m$ ), and the number of instruments ( $K_2$ ). A major restriction is that this test can be applied only if there are at least two overidentifying restrictions ( $K_2 + 2 \geq m$ ).

#### **Wald test distortion with i.i.d. errors**

The second approach considers the size distortion in the two-sided Wald statistic of  $\beta_1 = 0$ . For a single endogenous regressor, the true size of a nominal 5% significance level Wald test is

$$R = \Pr(|W_{2SLS}| > 1.96), \quad \text{where} \quad W_{2SLS} = \widehat{\beta}_{1,2SLS} / \sqrt{\widehat{V}(\widehat{\beta}_{1,2SLS})}$$

where  $R = 0.05$  under standard asymptotics but  $R > 0.05$  under weak-instrument asymptotics. We choose a tolerable level of size distortion, say, that true test size is at most 10%, and test at the 5% significance level the null hypothesis that  $R \leq 0.10$ . The critical values of  $F$  vary with  $R$ ,  $m$ , and  $K_2 \geq m$ .

#### **Stock and Yogo critical values for tests**

[Stock and Yogo \(2005\)](#) provided tables of critical values for the two tests under the assumption of i.i.d. errors. Test critical values vary with the

specified  $R$  or  $B$ , with the number of endogenous regressors ( $m$ ), and with the number of exclusion restrictions ( $K_2$ ). Stata commands presented below lead to output that includes these critical values.

The critical values are obtained using weak-instrument asymptotics, or local-to-zero asymptotics, that set first-stage coefficients  $\pi = \mathbf{c}/\sqrt{N}$ , where  $\mathbf{c}$  is a constant. Then the concentration parameter  $\tau^2$  in (7.15) remains constant as  $N \rightarrow \infty$ . The tests are conservative, leading to overrejection, because the distribution of both  $B^2$  and  $R$  additionally depend on  $\rho = \text{Cor}(u, v)$ , and to avoid this complication, one must compute test critical values using the value of  $\rho$  that maximizes  $B^2$  or that maximizes  $R$ .

For a single endogenous regressor, both tests reject if the  $F$  statistic is less than a critical value tabulated by [Stock and Yogo \(2005\)](#).

If there is more than one endogenous regressor, the relative bias  $B^2$  is defined with the quadratic forms in  $\beta_1$ , and  $R = \Pr \{ W_{2SLS} > \chi^2_{.05}(m) \}$ , where  $W_{2SLS} = \hat{\beta}_1' \{ \hat{V}(\hat{\beta}_1) \}^{-1} \hat{\beta}_1$ . The test statistic is then the Cragg–Donald minimum eigenvalue statistic. As noted in section 7.6.3, the minimum eigenvalue statistic was originally proposed by [Cragg and Donald \(1993\)](#) to test nonidentification. [Stock and Yogo \(2005\)](#) presume identification and interpret a low minimum eigenvalue to mean the instruments are weak. So the null hypothesis is that the instruments are weak against the alternative that they are strong.

#### **The F less than 10 rule of thumb**

In the case of a single endogenous regressor, the critical values for  $B = 0.10$  (a 10% maximum relative bias) range from 9.08 when  $K_2 = 3$  to 11.32 when  $K_2 = 30$ . And the critical values for a 5% Wald test with  $R = 0.10$  (a 10% maximum size distortion so true test size of at most 15%) range from 8.96 when  $K_2 = 1$  to 44.78 when  $K_2 = 30$ .

These critical values for what may be a tolerable amount of bias or test size distortion are generally more than 10. For models with a single endogenous regressor, this has led to a rule of thumb, one suggested by [Staiger and Stock \(1997\)](#), that if  $F < 10$ , then instruments are very likely to be weak and standard asymptotics should not be applied.

In practice, this rule is misinterpreted as meaning that if  $F > 10$ , then conventional Wald tests can be applied. There are many problems with this approach. In overidentified models, the Wald size-distortion test can have a critical value much larger than 10. The tests for weak instruments are restricted to models with i.i.d. errors, yet in practice microeconomics studies base inference on robust standard errors. And the size of a subsequent 5% Wald test of  $\beta_1$  is much higher than 5% because of pretesting for weak instruments; see section [7.7.1](#).

#### **Relative asymptotic bias of 2SLS with robust standard errors**

The preceding tests rely on the assumption of i.i.d. errors. This is very limiting, especially with clustered data, where there can be great difference between robust and nonrobust first-stage  $F$  statistics. [Montiel Olea and Pflueger \(2013\)](#) provide a relative-bias test that is valid for first-stage tests based on homoskedastic, heteroskedastic–robust, cluster–robust, or HAC standard errors. The test is restricted to the case of only one endogenous regressor and to 2SLS and LIML estimators, though this covers the majority of IV applications. It can be applied to both just-identified and overidentified models, whereas the Stock–Yogo relative-bias test requires at least two overidentifying restrictions.

Define  $F_{\text{eff}}$  to be the “effective”  $F$  statistic that varies with the method used to obtain the VCE of the estimator and whose formula is given in [Montiel Olea and Pflueger \(2013\)](#). In the just-identified case with robust standard errors,  $F_{\text{eff}}$  equals the first-stage  $F$  statistic computed using robust standard errors, while with homoskedastic errors  $F_{\text{eff}}$  reduces to the usual first-stage  $F$  statistic. More generally,  $F_{\text{eff}}$  differs from both the default  $F$  and the robust  $F$ .

Define  $\kappa$  to be the approximate bias of 2SLS (or of LIML), where the bias is calculated using a higher-order asymptotic approximation due to [Nagar \(1959\)](#), divided by a worst-case benchmark bias that occurs when instruments are completely uninformative and when the errors  $u$  and  $v$  are perfectly correlated. Then we obtain that value of  $F_{\text{eff}}$  for which

$$\Pr(|\kappa| > \kappa^*) = \alpha$$

for specified bias threshold  $\kappa^*$  and probability  $\alpha$ . The critical values generally vary with the dataset, so they are not available in a table. The community-contributed `weakivtest` postestimation command, presented below, provides the critical values for  $\kappa^* = 0.05, 0.10, 0.20$ , and  $0.30$  and for user-provided  $\alpha$  (with default  $\alpha = 0.05$ ).

In the special case of i.i.d. errors and nonrobust  $F$ , the critical values of this test with  $\kappa^* = 0.10$  and  $\alpha = 0.05$  are similar to the Stock–Yogo critical values for  $B = 0.10$  as  $K_2$  ranges from 3 to 30. [Andrews, Stock, and Sun \(2019\)](#) advocate use of  $F_{\text{eff}}$  for detecting weak instruments when errors are not i.i.d.

### 7.6.5 The `estat firststage` command

Following `ivregress`, various diagnostics and tests for weak instruments are provided by `estat firststage`. The syntax for this postestimation command is

```
estat firststage [ , all forceonnonrobust ]
```

The `all` option requests that all first-stage goodness-of-fit statistics be reported regardless of whether the model contains one or more endogenous regressors.

The `forceonnonrobust` option is used to allow use of `estat firststage` even when the preceding `ivregress` command computed a robust VCE. This has the advantage of computing the correct first-stage  $F$  statistic. But the Stock–Yogo critical values are no longer valid because they require i.i.d. errors. A better procedure, possible when there is a single endogenous regressor, is to use the community-contributed `weakivtest` postestimation command, which implements the effective  $F$  statistic test of [Montiel Olea and Pflueger \(2013\)](#); see section [7.6.7](#).

### 7.6.6 Just-identified model

We consider a just-identified model with one endogenous regressor, with `hi_empunion` instrumented by one variable, `ssiratio`. For completeness, we add the `all` option to print Shea's partial  $R^2$ , which is unnecessary here because we have only one endogenous regressor. Because we use `vce(robust)` in `ivregress`, we need to add the `forcenonrobust` option. The output is in three parts.

```
. * Weak instrument tests - just-identified with heteroskedastic-robust errors
. global x2list totchr age female blhisp linc
. qui ivregress 2sls ldrugexp (hi_empunion = ssiratio) $x2list, vce(robust)
. estat firststage, forcenonrobust all
```

First-stage regression summary statistics

Variable	Adjusted R-sq.	Partial R-sq.	Robust F(1,10061)	Prob > F
hi_empunion	0.0793	0.0787	0.0212	212.806 0.0000

Shea's partial R-squared

Variable	Shea's partial R-sq.	Shea's adj. partial R-sq.
hi_empunion	0.0212	0.0207

Minimum eigenvalue statistic = 217.963

Critical Values	# of endogenous regressors:	1
H0: Instruments are weak	# of excluded instruments:	1

2SLS relative bias	5%	10%	20%	30%
	(not available)			
2SLS size of nominal 5% Wald test	10%	15%	20%	25%
LIML size of nominal 5% Wald test	16.38	8.96	6.66	5.53

The first part is a summary table of key diagnostic statistics that are useful in suggesting weak instruments. The first two statistics are the  $R^2$  and adjusted- $R^2$  from the first-stage regression. These are around 0.08, so there will be considerable loss of precision because of IV estimation. They are not low enough to flag a weak-instruments problem, although, as already noted, there may still be a problem because `ssiratio` may be contributing very little to this fit. To isolate the explanatory power of `ssiratio` in explaining

`hi.empunion`, two statistics are given. The partial  $R^2$  is that between `hi.empunion` and `ssiratio` after controlling for `totchr`, `age`, `female`, `blhisp`, and `linc`. This is quite low at 0.0212, suggesting some need for caution. The final statistic is an  $F$  statistic for the joint significance of the instruments excluded from the structural model. For this just-identified example, this is a test on just `ssiratio`, and  $F = 212.81$ , based on heteroskedastic–robust standard errors, is simply the square of the  $t$  statistic from the first-stage regression ( $14.66^2 = 212.81$ ). This  $F$  statistic of 212.81 is considerably larger than the rule of thumb value of 10 that is sometimes suggested, so `ssiratio` does not seem to be a weak instrument.

The second part gives Shea’s partial  $R^2$ , which equals the previously discussed partial  $R^2$  because there is just one endogenous regressor.

The third part implements the tests of [Stock and Yogo \(2005\)](#). As already noted, their use here is questionable because robust standard errors have been used. The relative bias test is not available, because the model is just identified, rather than overidentified by two or more restrictions. The output for the Wald test distortion test gives critical values for both the 2SLS estimator and the LIML estimator. We are using the 2SLS estimator, which is equivalent to the LIML estimator in the just-identified case. If we are willing to tolerate distortion for a 5% Wald test based on the 2SLS estimator, so that the true size can be at most 10%, then we reject the null hypothesis of weak instruments because  $F = 212.81 > 16.38$ .

The Stock–Yogo critical values are not applicable here. In this case, however, default standard errors lead to a first-stage  $F = 217.96$ , which is similar to the heteroskedastic–robust  $F = 212.81$ , and both  $F$  statistics greatly exceed 16.38, so we feel comfortable in rejecting the null hypothesis of weak instruments. In other cases, use of `estat firststage` with robust standard errors will be much more questionable.

### 7.6.7 The `weakivtest` command

The community-contributed `weakivtest` command ([Pflueger and Wang 2015](#)) implements the relative asymptotic bias test of [Montiel Olea and Pflueger \(2013\)](#) following the `ivregress` command (or the `ivreg2` command, which is presented in section [7.6.9](#)).

As an example, we repeat the preceding just-identified example with heteroskedastic-robust standard errors. We obtain

```
. * Weak instrument tests - just-identified using weakivtest
. qui ivregress 2sls ldrugexp (hi_empunion = ssiratio)
> $x2list, vce(robust)
. weakivtest
(obs=10,068)
```

Montiel-Pflueger robust weak instrument test

---

Effective F statistic:	212.806
Confidence level alpha:	5%

---

Critical Values	TSLS	LIML
% of Worst Case Bias		
tau=5%	37.418	37.418
tau=10%	23.109	23.109
tau=20%	15.062	15.062
tau=30%	12.039	12.039

---

The test is being performed at the 5% confidence level; this is an option that can be changed. The effective  $F$  statistic in this just-identified example equals the nonrobust  $F$  statistic of 212.81. The critical value for 2SLS with 10% worst case bias is 23.109, compared with, for example, 16.38 for Wald test size distortion of 5% using Stock–Yogo critical values that assume i.i.d. errors. Despite this higher threshold, we reject the null hypothesis of weak instruments.

### 7.6.8 Overidentified model

For a model with a single endogenous regressor that is overidentified, the output is of the same format as the previous example. The  $F$  statistic will now be a joint test for the several instruments. If there are three or more instruments, so that there are two or more overidentifying restrictions, then the Stock–Yogo 2SLS relative-bias criterion can be used.

We consider an example with three overidentifying restrictions:

```

. * Weak instrument tests - overidentified with heteroskedastic-robust errors
. qui ivregress 2sls ldrugexp (hi_empunion = ssiratio lowincome multlc firmsz)
>      $x2list, vce(robust)
. estat firststage, forcenonrobust

```

First-stage regression summary statistics

Variable	Adjusted R-sq.	Partial R-sq.	Robust F(4,10058)	Prob > F
hi_empunion	0.0846	0.0838	0.0269	69.8112 0.0000

Minimum eigenvalue statistic = 69.4773

Critical Values	# of endogenous regressors:	1
H0: Instruments are weak	# of excluded instruments:	4

	5%	10%	20%	30%
2SLS relative bias	16.85	10.27	6.71	5.34
2SLS size of nominal 5% Wald test	10%	15%	20%	25%
LIML size of nominal 5% Wald test	24.58	13.96	10.26	8.31
	5.44	3.87	3.30	2.98

Using either  $F = 69.81$  or the minimum eigenvalue of 69.48, we firmly reject the null hypothesis of weak instruments. The endogenous regressor `hi_empunion`, from structural-model estimates not given, has a coefficient of  $-0.817$  and a standard error of 0.170 compared with  $-0.811$  and 0.188 when `ssiratio` is the only instrument.

The Stock–Yogo critical values for both tests are not appropriate here. From output not given, the `weakivtest` command yields an effective  $F$  equal to 74.36 that exceeds 5% test-level critical values of 19.92 and 12.03 for, respectively, Wald test bias of 5% or 10%. The instruments do not appear to be weak.

### 7.6.9 The `ivreg2` command

The community-contributed `ivreg2` command (see [Baum, Schaffer, and Stillman \[2007\]](#)) provides additional estimators and tests to those provided by the `ivregress` command and stores many results conveniently in `e()`. It includes 2SLS, LIML, GMM,  $k$ -class, and continuously updating estimators. It computes a range of robust standard errors, including two-way cluster-robust standard errors. It provides endogeneity tests. And it provides a wider

range of weak-instrument tests and diagnostics. Their 2007 article provides a very good overview of IV methods and of the `ivreg2` command.

The syntax for the `ivreg2` command is similar to that for the `ivregress` command. The `first`, `sfirst`, and `ffirst` options provide varying amounts of detail from the first-stage regressions.

We fit the model with four instruments for the single endogenous regressors, use the `first` option, and request heteroskedastic–robust standard errors. We obtain

```
. * ivreg2 for overidentified model with heteroskedastic-robust standard errors
. ivreg2 ldrugexp (hi_empunion = ssiratio lowincome multlc firmsz) $x2list,
>     robust first
First-stage regressions
```

---

First-stage regression of hi\_empunion:

Statistics robust to heteroskedasticity

Number of obs = 10068

hi_empunion	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
ssiratio	-.1962855	.0153985	-12.75	0.000	-.2264696	-.1661015
lowincome	-.0586565	.0118359	-4.96	0.000	-.0818573	-.0354556
multlc	.1104924	.0208033	5.31	0.000	.0697139	.151271
firmsz	.0035953	.0018985	1.89	0.058	-.0001261	.0073168
totchr	.0137979	.0036355	3.80	0.000	.0066716	.0209243
age	-.0078626	.0007087	-11.09	0.000	-.0092518	-.0064734
female	-.07128	.0096042	-7.42	0.000	-.0901062	-.0524538
blhisp	-.0658391	.0121769	-5.41	0.000	-.0897082	-.0419699
linc	.0387941	.0062042	6.25	0.000	.0266326	.0509556
_cons	1.000387	.0578014	17.31	0.000	.8870848	1.113689

F test of excluded instruments:

F( 4, 10058) = 69.81

Prob > F = 0.0000

Sanderson-Windmeijer multivariate F test of excluded instruments:

F( 4, 10058) = 69.81

Prob > F = 0.0000

---

Summary results for first-stage regressions

---

Variable	(Underid)			(Weak id)		
	F( 4, 10058)	P-val	SW Chi-sq( 4)	P-val	SW F( 4, 10058)	
hi_empunion	69.81	0.0000	279.52	0.0000	69.81	

NB: first-stage test statistics heteroskedasticity-robust

Stock-Yogo weak ID F test critical values for single endogenous regressor:

5% maximal IV relative bias	16.85
10% maximal IV relative bias	10.27
20% maximal IV relative bias	6.71
30% maximal IV relative bias	5.34
10% maximal IV size	24.58
15% maximal IV size	13.96
20% maximal IV size	10.26
25% maximal IV size	8.31

Source: Stock-Yogo (2005). Reproduced by permission.

NB: Critical values are for i.i.d. errors only.

Underidentification test

Ho: matrix of reduced form coefficients has rank=K1-1 (underidentified)

Ha: matrix has rank=K1 (identified)

Kleibergen-Paap rk LM statistic Chi-sq(4)=234.51 P-val=0.0000

Weak identification test

Ho: equation is weakly identified

Cragg-Donald Wald F statistic 69.48

Kleibergen-Paap Wald rk F statistic 69.81

Stock-Yogo weak ID test critical values for K1=1 and L1=4:

5% maximal IV relative bias	16.85
10% maximal IV relative bias	10.27
20% maximal IV relative bias	6.71
30% maximal IV relative bias	5.34
10% maximal IV size	24.58
15% maximal IV size	13.96
20% maximal IV size	10.26
25% maximal IV size	8.31

Source: Stock-Yogo (2005). Reproduced by permission.

NB: Critical values are for Cragg-Donald F statistic and i.i.d. errors.

Weak-instrument-robust inference

Tests of joint significance of endogenous regressors B1 in main equation

Ho: B1=0 and orthogonality conditions are valid

Anderson-Rubin Wald test F(4,10058)= 9.74 P-val=0.0000

Anderson-Rubin Wald test Chi-sq(4)= 38.99 P-val=0.0000

Stock-Wright LM S statistic Chi-sq(4)= 35.94 P-val=0.0000

NB: Underidentification, weak identification and weak-identification-robust test statistics heteroskedasticity-robust

Number of observations N = 10068  
Number of regressors K = 7  
Number of endogenous regressors K1 = 1  
Number of instruments L = 10  
Number of excluded instruments L1 = 4

IV (2SLS) estimation

---

Estimates efficient for homoskedasticity only

Statistics robust to heteroskedasticity

Total (centered) SS = 18664.39028	Number of obs = 10068
Total (uncentered) SS = 441571.929	F( 6, 10061) = 339.04
Residual SS = 17137.60036	Prob > F = 0.0000
	Centered R2 = 0.0818
	Uncentered R2 = 0.9612
	Root MSE = 1.305

ldrugexp	Coefficient	Robust			
		std. err.	z	P> z	[95% conf. interval]
hi_empunion	-.8174208	.1705701	-4.79	0.000	-1.151732 -.4831095
totchr	.4492737	.0100416	44.74	0.000	.4295926 .4689548
age	-.0123008	.0026668	-4.61	0.000	-.0175276 -.0070741
female	-.0145257	.0302726	-0.48	0.631	-.073859 .0448075
blhisp	-.2093601	.0380957	-5.50	0.000	-.2840264 -.1346939
linc	.0820149	.0197741	4.15	0.000	.0432584 .1207714
_cons	6.698209	.2335101	28.68	0.000	6.240537 7.15588

---

Underidentification test (Kleibergen-Paap rk LM statistic): 234.509  
Chi-sq(4) P-val = 0.0000

Weak identification test (Cragg-Donald Wald F statistic):	69.477
(Kleibergen-Paap rk Wald F statistic):	69.811
Stock-Yogo weak ID test critical values:	
5% maximal IV relative bias	16.85
10% maximal IV relative bias	10.27
20% maximal IV relative bias	6.71
30% maximal IV relative bias	5.34
10% maximal IV size	24.58
15% maximal IV size	13.96
20% maximal IV size	10.26
25% maximal IV size	8.31

Source: Stock-Yogo (2005). Reproduced by permission.

NB: Critical values are for Cragg-Donald F statistic and i.i.d. errors.

---

Hansen J statistic (overidentification test of all instruments):	11.027
Chi-sq(3) P-val =	0.0116

---

Instrumented: hi\_empunion

Included instruments: totchr age female blhispl linc

Excluded instruments: ssiratio lowincome multlc firmsz

---

The first set of output presents OLS estimates of the first-stage model for the endogenous regressor `hi_empunion`, and the output is identical to that obtained using `ivregress` with option `first`. If we had used option `sfirst` rather than `first`, the output would additionally include OLS estimates of the similar first-stage model for `ldrugexp`.

The next set of output gives various summary measures from the first-stage regression. The  $F = 69.81$  is the same as that given in section [7.6.8](#) using the `estat firststage` command, and the Cragg–Donald statistic of 69.48 is the  $F$  statistic if default nonrobust standard errors were used.

Critical values are available for both Stock–Yogo weak-instrument tests because there are at least two overidentifying restrictions, though these are for models with i.i.d. errors. There is no indication of a weak-instrument problem. The output also gives the Anderson–Rubin Wald test for statistical significance of the endogenous regressor `hi_empunion` that is robust to weak instruments as explained in section [7.7.2](#).

The final set of output gives 2SLS estimates of the structural equation that equal those from the `ivregress` command given in section [7.4.4](#). It includes some repetition of the underidentification and weak identification tests because the final set of output is all that would be given if the `first` option was not used. The overidentification test statistic equals 11.03 with  $p = 0.012$ , so the three overidentifying restrictions are rejected at level 0.05.

The endogenous regressor `hi_empunion` has  $z = -4.79$ , corresponding to  $F(1, N - K) = 4.79^2 = 22.94$ . By comparison, the Anderson–Rubin statistic, which is robust to weak instruments, has  $F(4, N - K) = 9.74$ . In both cases,  $p = 0.000$ , so the endogenous variable `hi_empunion` is statistically significant at the 5% level.

### 7.6.10 More than one endogenous regressor

The preceding applications had just one endogenous regressor. With more than one endogenous regressor, `estat firststage` reports weak-instruments diagnostics that include Shea’s partial  $R^2$  and the Stock–Yogo tests based on the Cragg–Donald minimum eigenvalue statistic, which generalizes the  $F$  statistic. The `all` option additionally leads to reporting for each endogenous regressor the first-stage regression and associated  $F$  statistic and partial  $R^2$ .

The Cragg–Donald minimum eigenvalue statistic provides an overall test of weak instruments. [Sanderson and Windmeijer \(2016\)](#) propose corrected conditional  $F$  statistics for the first-stage regressions of the endogenous regressors that, under the assumption of i.i.d. errors, can be compared with Stock–Yogo critical values. This provides additional detail on the nature of any weak-instrument problem. The `ivreg2` output includes the Sanderson–Windmeijer test.

### 7.6.11 Sensitivity to choice of instruments

In the main equation, `hi_empunion` has a strong negative impact on `ldrugexp`. This contrasts with a small positive effect observed in the OLS results when `hi_empunion` is treated as exogenous; see section [7.4.6](#). If our instrument `ssiratio` is valid, then this would suggest a substantial bias in the OLS result. But is this result sensitive to the choice of the instrument?

To address this question, we compare results for four just-identified specifications, each estimated using just one of the four available instruments. We present a table with the structural-equation estimates for OLS and for the four IV estimations, followed by the heteroskedastic–robust first-stage  $F$  statistic. We use the `ivreg2` command because it actually saves the  $F$  statistic in `e(widstat)`. We have

```

. * Compare four just-identified model estimates with different instruments
. qui regress ldrugexp hi_empunion $x2list, vce(robust)
. estimates store OLS0
. qui ivreg2 ldrugexp (hi_empunion=ssiratio) $x2list, robust
. estimates store IV_INST1
. scalar f1 = e(widstat)
. qui ivreg2 ldrugexp (hi_empunion=lowincome) $x2list, robust
. estimates store IV_INST2
. scalar f2 = e(widstat)
. qui ivreg2 ldrugexp (hi_empunion=multlc) $x2list, robust
. estimates store IV_INST3
. scalar f3 = e(widstat)
. qui ivreg2 ldrugexp (hi_empunion=firmsz) $x2list, robust
. estimates store IV_INST4
. scalar f4 = e(widstat)

. estimates table OLS0 IV_INST1 IV_INST2 IV_INST3 IV_INST4, b(%8.4f) se

```

Variable	OLS0	IV_INST1	IV_INST2	IV_INST3	IV_INST4
hi_empunion	0.0732 0.0260	-0.8115 0.1884	0.0910 0.3607	-1.3491 0.4249	-2.9340 1.4038
totchr	0.4402 0.0094	0.4492 0.0100	0.4400 0.0101	0.4547 0.0116	0.4708 0.0204
age	-0.0034 0.0019	-0.0122 0.0028	-0.0032 0.0042	-0.0176 0.0048	-0.0335 0.0144
female	0.0570 0.0254	-0.0141 0.0310	0.0584 0.0382	-0.0572 0.0449	-0.1844 0.1201
blhisp	-0.1487 0.0342	-0.2090 0.0384	-0.1475 0.0417	-0.2455 0.0490	-0.3534 0.1099
linc	0.0116 0.0137	0.0815 0.0208	0.0102 0.0312	0.1240 0.0373	0.2492 0.1135
_cons	5.8464 0.1574	6.6925 0.2449	5.8295 0.3835	7.2067 0.4443	8.7224 1.3651

Legend: b/se

```

. display "Robust first-stage F: " f1 _s(2) f2 _s(2) f3 _s(2) f4
Robust first-stage F: 212.80621 58.760623 52.412092 14.357187

```

The different instruments produce very different IV estimates for the coefficient of the endogenous regressor `hi_empunion`, though they are within two standard errors of each other (with the exception of that with `lowincome` as the instrument). All differ greatly from OLS estimates, aside from when

`lowincome` is the instrument (`IV_INST2`). The coefficient of the most highly statistically significant regressor, `totchr`, changes little with the choice of instrument. Coefficients of some of the other exogenous regressors change considerably, though there is no sign reversal aside from `female`.

The heteroskedastic–robust first-stage  $F$  statistic for the final model with `firmsz` is low enough to indicate a weak-instrument problem using this instrument. Because inference here is heteroskedastic–robust, we perform the relative asymptotic bias test of [Montiel Olea and Pflueger \(2013\)](#), with maximum relative 10% bias occurring with probability 0.05. We obtain

```
. * Montiel-Olea critical value for 10% relative bias with alpha = 0.05
. qui ivregress 2sls ldrugexp (hi_empunion=firmsz) $x2list, vce(robust)
. qui weakivtest
. di "Effective F = " r(F_eff) " with critical value = " r(c_TSLS_10)
Effective F = 14.357187 with critical value = 23.10851
. clear mata
```

The final instrument, `firmsz`, is weak.

When we perform a similar sensitivity analysis by progressively adding `lowincome`, `multlc`, and `firmsz` as instruments to an originally just-identified model with `ssiratio` as the instrument, there is little change in the 2SLS estimates; see the exercises at the end of this chapter.

There are several possible explanations for the sensitivity in the just-identified case. The results could reflect the expected high variability of the IV estimator in a just-identified model, variability that also reflects the differing strength of different instruments. Some of the instruments may not be valid instruments. If treatment effects are heterogeneous, then IV can be given a LATE interpretation (see section 25.5), especially in the two cases (`lowincome` and `multlc`) with binary instruments. Then the instruments may correspond to different LATES.

Perhaps it is equally the case that the results reflect model misspecification—relative to a model one would fit in a serious empirical analysis of `ldrugexp`, the model used in the example is rather simple. While here we concentrate on the statistical tools for exploring the issue, in practice

a careful context-specific investigation based on relevant theory is required to satisfactorily resolve the issue.

## 7.7 Inference with weak instruments

The preceding section presented diagnostics and tests for weak instruments. Here we first discuss problems that arise from the common practice of performing hypothesis tests and computing confidence intervals following pretesting for weak instruments.

We then present inference that is valid regardless of whether instruments are weak and does not entail pretesting for weak instruments. The leading example is the Anderson–Rubin Wald test, the preferred method in a just-identified model.

### 7.7.1 Tests following pretesting for weak instruments

A common procedure is to first screen for weak instruments and then perform regular asymptotic inference on the coefficient of endogenous regressors if the test is passed. Such pretesting is problematic because it distorts test size.

For simplicity, consider the leading case of a just-identified equation with a single endogenous regressor. We decide that instruments are not weak if the first-stage  $F$  exceeds some critical value  $F^*$ , such as  $F^* = 10$ , and then proceed to perform a regular two-sided Wald test of  $H_0: \beta_1 = 0$  and determine statistical significance at 5% if  $|t| > 1.96$ .

[Andrews, Stock, and Sun \(2019\)](#), for example, provide Monte Carlo evidence that such screening on the first-stage  $F$  can lead to large size distortion of the subsequent Wald test. The problem is that the true test size is not simply  $\Pr(|t| > 1.96)$  but is instead  $\Pr\{(F > F^*) \cup (|t| > 1.96)\}$ . For example, with i.i.d. errors, the Stock–Yogo critical value for a 5% Wald test with size distortion of at most 10% is 8.96. So if we pass a weak-instrument screening test of whether  $F > 8.96$ , then the true test size of a nominal 5% test of whether  $\beta_1 = 0$  could be as high as 15%!

[Lee et al. \(2021\)](#) argue that weak instruments can be a problem in some applications that have  $F$  much, much larger than 10. They address this issue

in detail for the model with single endogenous regressor and single instrument. They use weak-instrument asymptotics that permit non-i.i.d. errors, such as clustered errors, provided the  $F$  statistic and Wald test are based on the same robust standard error method. The authors find that a nominal 5% Wald test with critical value 1.96 that follows a pretest that  $F > 10$  has true size that could be as high as 11.3%. Even more problematic is that it is very difficult to attain a test with true size of 5%. Without other restrictions, it is possible if  $F > 104.7$  and  $|t| > 1.96$ , which requires a very strong first stage, and it is possible if  $F > 10$  and  $|t| > 3.43$ , which requires a highly statistically significant structural parameter estimate. The authors provide tables of selected values of adjusted Wald test critical values, called  $t_F$  critical values, that decrease as the first-stage  $F$  statistic increases.

These limitations become even greater when effective sample sizes are not so large that asymptotic theory provides a good approximation; see section [7.8](#).

If it is felt that instruments are likely to be weak, it is best to not first screen on the first-stage  $F$  statistic, though it remains informative to report  $F$ . Instead, one should use inference methods that are robust to weak instruments, methods that we now present.

### 7.7.2 Anderson–Rubin Wald test

Consider the model with structural equation [\(7.5\)](#) and first-stage equations [\(7.6\)](#). In matrix terms,  $\mathbf{y}_1 = \mathbf{Y}_2\boldsymbol{\beta}_1 + \mathbf{X}_1\boldsymbol{\beta}_2 + \mathbf{u}$  and  $\mathbf{Y}_2 = \mathbf{X}_1\Pi_1 + \mathbf{X}_2\Pi_2 + \mathbf{V}$ , where  $\mathbf{X}_2$  are the additional instruments for the endogenous regressors  $\mathbf{y}_2$ .

The reduced-form equation for  $\mathbf{y}_1$  is obtained by substituting out  $\mathbf{Y}_2$ . We have

$$\begin{aligned}\mathbf{y}_1 &= \mathbf{Y}_2\boldsymbol{\beta}_1 + \mathbf{X}_1\boldsymbol{\beta}_2 + \mathbf{u} \\ &= (\mathbf{X}_1\Pi_1 + \mathbf{X}_2\Pi_2 + \mathbf{V})\boldsymbol{\beta}_1 + \mathbf{X}_1\boldsymbol{\beta}_2 + \mathbf{u} \\ &= \mathbf{X}_1(\Pi_1\boldsymbol{\beta}_1 + \boldsymbol{\beta}_2) + \mathbf{X}_2(\Pi_2\boldsymbol{\beta}_1) + (\mathbf{V}\boldsymbol{\beta}_1 + \mathbf{u}) \\ &= \mathbf{X}_1\boldsymbol{\gamma} + \mathbf{X}_2\boldsymbol{\alpha} + \mathbf{w}\end{aligned}$$

where  $\alpha = \Pi_2 \times \beta_1$ . For a just-identified single endogenous regressor model,  $\alpha = \pi_2 \beta_1$ .

It follows that  $\beta_1 = \mathbf{0}$  implies that  $\alpha = \mathbf{0}$ . So we can test  $H_0: \beta_1 = \mathbf{0}$  by performing a Wald test of whether the coefficients of the instruments  $\mathbf{x}_{2i}$  are 0 in the reduced-form regression of  $y_{1i}$  on the exogenous regressors  $\mathbf{x}_{1i}$  and the instruments  $\mathbf{x}_{2i}$ .

The attraction of this approach is that the right-hand side regressors do not involve the endogenous regressors  $\mathbf{y}_{2i}$ . This leads to a denominator in the formula for the estimator that does not have the randomness property that causes problems for the usual 2SLS estimator. So there is no problem of weak instruments, and one can use regular asymptotics. This test due to [Anderson and Rubin \(1949\)](#) is called the AR test. As pointed out by [Chernozhukov and Hansen \(2008\)](#), one can also base inference on heteroskedastic–robust, cluster–robust, or HAC standard errors.

The limitation of this approach is that it loses power in overidentified models. One way to see this is to note that in the common case of a single endogenous regressor, we want to test only the scalar  $\beta_1$ , but the test is a joint test of  $K_2$  parameters, where  $K_2$  is the number of instruments  $\mathbf{x}_{2i}$ . A second way to see this is to note that  $\alpha$  also equals 0 if  $\Pi_2 = \mathbf{0}$ , so the test is also one of identification.

#### Anderson–Rubin test example of test of statistical significance

To illustrate the method, we consider an overidentified example, with instruments `ssiratio` and `lowincome` for the single endogenous regressor `hi_empunion`, so  $\beta_1$  is a scalar, in the structural equation with dependent variable `1drugexp`. Heteroskedastic-robust inference is performed. We have

```
. * Anderson-Rubin test for overidentified sample with robust standard errors
. regress ldrugexp ssiratio firmsz $x2list, vce(robust)
```

Linear regression

Number of obs	=	10,068
F(7, 10060)	=	326.22
Prob > F	=	0.0000
R-squared	=	0.1788
Root MSE	=	1.2343

ldrugexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
ssiratio	.1740745	.0395217	4.40	0.000	.096604	.251545
firmsz	-.0191037	.0059784	-3.20	0.001	-.0308226	-.0073847
totchr	.4382028	.0093954	46.64	0.000	.4197859	.4566196
age	-.0055112	.0019412	-2.84	0.005	-.0093163	-.0017061
female	.0451868	.0253462	1.78	0.075	-.0044968	.0948704
blhisp	-.1572397	.0341391	-4.61	0.000	-.2241591	-.0903202
linc	.0449053	.0148784	3.02	0.003	.0157407	.0740699
_cons	5.865519	.1557835	37.65	0.000	5.560152	6.170886

```
. test ssiratio firmsz
( 1) ssiratio = 0
( 2) firmsz = 0
F( 2, 10060) =    14.98
Prob > F =      0.0000
```

The Anderson–Rubin Wald test statistic for statistical significance of `hi_empunion` is  $F = 14.98$  with  $p = 0.0000$ , so the endogenous regressor is highly statistically significant. This coincides with the value found using the `ivreg2` command in section [7.6.9](#).

### Anderson–Rubin test for general test

The AR procedure can be used to test values of  $\beta_1$  other than zero.

Suppose we wish to test  $H_0: \beta_1 = \beta_{10}$ . If we subtract  $\mathbf{Y}_2\beta_{10}$  from both sides of the original structural equation and obtain a reduced form as before, we have

$$\begin{aligned} \mathbf{y}_1 - \mathbf{Y}_2\beta_{10} &= \mathbf{Y}_2(\beta_1 - \beta_{10}) + \mathbf{X}_1\beta_2 + \mathbf{u} \\ &= (\mathbf{X}_1\Pi_1 + \mathbf{X}_2\Pi_2 + \mathbf{V})(\beta_1 - \beta_{10}) + \mathbf{X}_1\beta_2 + \mathbf{u} \\ &= \mathbf{X}_1\{\Pi_1(\beta_1 - \beta_{10}) + \beta_2\} + \mathbf{X}_2\{\Pi_2(\beta_1 - \beta_{10})\} + \{\mathbf{V}(\beta_1 - \beta_{10}) + \mathbf{u}\} \\ &= \mathbf{X}_1\gamma + \mathbf{X}_2\alpha + \mathbf{w} \end{aligned}$$

where now  $\alpha = \Pi_2 \times (\beta_1 - \beta_{10})$ . Thus, we perform the same reduced-form regression as before, except the dependent variable is now  $y_{1i} - \mathbf{y}'_{2i}\beta_{10}$ .

The next example does so for a test of  $H_0: \beta_1 = -0.6$ .

```
. * Anderson-Rubin test for beta = -0.6 rather than beta = 0
. qui generate yfordiffbeta = ldrugexp - (-0.6)*hi_empunion
. regress yfordiffbeta ssiratio firmsz $x2list, vce(robust)

Linear regression
Number of obs      =      10,068
F(7, 10060)        =     312.39
Prob > F          =     0.0000
R-squared          =     0.1724
Root MSE           =     1.2754
```

yfordiffbeta	Coefficient	Robust			
		std. err.	t	P> t	[95% conf. interval]
ssiratio	.0426252	.0406361	1.05	0.294	-.0370298 .1222802
firmsz	-.0157014	.0065527	-2.40	0.017	-.0285459 -.0028569
totchr	.4462441	.0097066	45.97	0.000	.4272172 .4652709
age	-.010578	.0020055	-5.27	0.000	-.0145091 -.0066469
female	.0017949	.0262003	0.07	0.945	-.049563 .0531528
blhisp	-.1942316	.0352794	-5.51	0.000	-.2633862 -.1250769
linc	.0721462	.0154753	4.66	0.000	.0418116 .1024809
_cons	6.486113	.1611032	40.26	0.000	6.170319 6.801908

```
. test ssiratio firmsz
( 1) ssiratio = 0
( 2) firmsz = 0
F( 2, 10060) =     3.44
Prob > F =     0.0320
```

The hypothesis that  $\beta = -0.6$  is rejected at level 0.05 because  $p = 0.032 < 0.05$ .

### 7.7.3 Anderson–Rubin Wald confidence regions

From section 11.3.13, a confidence interval, or more generally a confidence region, can be obtained by inverting a test. Specifically, to obtain a 95% confidence set for a parameter  $\theta$ , we perform a two-sided test of  $\theta = \theta^*$  for a range of values of  $\theta^*$ . The confidence set is then those values of  $\theta^*$  for which

the test has  $p > 0.05$  because the 95% confidence interval includes those values that we do not reject at level 0.05.

In the single endogenous regressor case, we fit by OLS the model  $y_{1i} - \beta_1^* y_{2i} = \mathbf{x}'_{1i} \gamma + \mathbf{x}'_{2i} \alpha + w_i$  for a range of values of  $\beta_1^*$ . A 95% Wald AR confidence region for  $\beta_1$  is those values of  $\beta_1^*$  for which a Wald test at level 5% does not lead to rejection of the hypothesis that  $\alpha = 0$ .

For the current example, we know from the preceding AR 5% tests that a 95% confidence interval does not include  $\beta_1 = 0.0$ , because  $H_0: \beta_1 = 0.0$  was rejected at level 0.05. Similarly, it does not include  $\beta_1 = -0.6$ , because in section [7.7.2](#)  $H_0: \beta_1 = -0.06$  was not rejected at level 0.05. From output given below, the community-contributed `condivreg` command yields a heteroskedastic-robust 95% confidence interval  $[-1.045, -0.791]$  for the current overidentified example.

In general, inversion of tests can yield confidence intervals that are disjoint, such as  $\{(a, b) \cup (c, d)\}$ , or even empty. Thus, the confidence intervals are more correctly referred to as confidence regions or confidence sets.

The AR confidence regions could be any of the forms 1)  $(b_1, b_2)$ ; 2)  $(-\infty, b_1) \cup (b_2, \infty)$ ; 3)  $(-\infty, \infty)$ ; or 4)  $\emptyset$ , where  $\emptyset$  denotes the empty set. The third case can arise when the instruments are so weak that the data provide very little information on  $\beta_1$ . The fourth case can arise because the AR test is also one of identification.

Practitioners can find this unsettling because the standard Wald test based on  $\widehat{\beta}_{1,2SLS}$  always yields confidence regions of the first form. The AR Wald test based on  $\widehat{\alpha}$  has the added complication that the estimate of  $\alpha$  varies with  $\beta_1^*$ . In the just-identified case, we denote the estimate  $\widehat{\alpha}(\beta_1^*)$ . Then the nonrejection region is  $[\widehat{\alpha}(\beta_1^*)/\text{se}\{\widehat{\alpha}(\beta_1^*)\}]^2 < 1.96^2$ .

#### 7.7.4 Tests under weak-instrument asymptotics

The Anderson–Rubin approach does not require the use of weak-instrument asymptotics. It has low power in overidentified models, however, leading to

alternative procedures that do use weak asymptotics.

We focus on inference on the coefficients  $\beta_1$  of the endogenous regressors in the structural model. Then weak-instrument asymptotics yield tests with asymptotically correct size and confidence intervals with asymptotically correct coverage even if instruments are weak.

Specifically, the asymptotics assume that  $\Pi_2 = \mathbf{C}/\sqrt{N}$ , where  $\Pi_2$  are the coefficients of the instruments in the first-stage model for the endogenous regressors:  $\mathbf{Y}_2 = \mathbf{X}_1\Pi_1 + \mathbf{X}_2\Pi_2 + \mathbf{V}$ . Specifying  $\Pi_2$  to decline at rate  $\sqrt{N}$  has the consequence that the concentration parameter in (7.15) is constant as the sample size grows, rather than declining at rate  $1/N$ .

The analysis is complicated by the fact that, unlike conventional asymptotics, the Wald, LR, and LM weak-instrument tests are not asymptotically equivalent under local alternatives if the model is overidentified.

#### Conditional likelihood-ratio test

For models with i.i.d. errors, [Moreira \(2003\)](#) proposed the conditional likelihood-ratio (CLR) test. The test is the LR test for  $H_0: \beta_1 = \beta_{10}$  based on the assumptions that errors in the reduced form for  $(y_1, y_2)$  are i.i.d.  $N(\mathbf{0}, \Omega)$  and that  $\Omega$  is known; the term “conditional” arises because it is the LR test conditional on  $\Omega$  being known. The resulting properties of the test require only the i.i.d. assumption; they do not require normality or that  $\Omega$  be known.

With a single endogenous regressor and a just-identified model, the test reduces to the AR Wald test. In overidentified models, the CLR test has optimal power and has better power than an LM test, while the AR test can have very poor power.

#### The condivreg command

The community-contributed `condivreg` command, developed by [Mikusheva and Poi \(2006\)](#), implements these tests that are surveyed and further

developed in [Andrews, Moreira, and Stock \(2007\)](#) and [Mikusheva \(2010\)](#). It also reports confidence intervals that are obtained by inverting the test statistics.

The `condivreg` command has the same basic syntax as `ivregress`, except that the specific estimator used (`2sls` or `liml`) is passed as an option. The default is to report a corrected  $p$ -value for the test of statistical significance and a corrected 95% confidence region based on the CLR test statistic ([Moreira 2003](#)). The `lm` option computes the LM test and associated confidence interval, and the `ar` option computes the Anderson–Rubin test statistic. The `test(#)` option is used to get  $p$ -values for tests of values other than zero for the coefficient of the endogenous regressor.

For the current overidentified example with two instruments, we have

```
. * Condivreg: Weak instrument robust inference for i.i.d. errors
. condivreg ldrugexp (hi_empunion=ssiratio firmsz) $x2list, 2sls lm ar test(0)
Instrumental variables (2SLS) regression
```

First-stage results

F( 2, 10060) = 112.55	Number of obs = 10068
Prob > F = 0.0000	F( 6, 10061) = 319.93
R-squared = 0.0799	Prob > F = 0.0000
Adj R-squared = 0.0793	R-squared = 0.0654
	Adj R-squared = 0.0649
	Root MSE = 1.317

ldrugexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
hi_empunion	-.8910603	.1881943	-4.73	0.000	-1.259959 -.5221619
totchr	.4500233	.0103835	43.34	0.000	.4296695 .4703771
age	-.0130392	.0027449	-4.75	0.000	-.0184197 -.0076586
female	-.0204353	.030708	-0.67	0.506	-.0806291 .0397584
blhisp	-.214372	.0382284	-5.61	0.000	-.2893072 -.1394368
linc	.0878329	.0209322	4.20	0.000	.0468016 .1288642
_cons	6.768635	.2416588	28.01	0.000	6.294936 7.242335

Instrumented: hi\_empunion

Instruments: totchr age female blhisp linc ssiratio firmsz

Confidence set and p-value for hi\_empunion are based on normal approximation

---

Coverage-corrected confidence sets and p-values  
for Ho:  $\_b[hi\_empunion] = 0$   
LIML estimate of  $\_b[hi\_empunion] = -.91599$

Test	Confidence Set	p-value
Conditional LR	[-1.313162, -.5549692]	0.0000
Anderson-Rubin	[-1.045006, -.7910382]	0.0000
Score (LM)	[-1.317044, -.5517649] U [ 6.650289, 7.420294]	0.0000

All tests have  $p = 0.0000$  and strongly reject the null hypothesis that hi\_empunion is statistically insignificant.

[Mikusheva \(2010\)](#) develops fast algorithms for computing the tests and summarizes the possible shapes of the AR, CLR, and LM confidence regions when errors are i.i.d.

The AR confidence regions could be any of the forms 1)  $(b_1, b_2)$ ; 2)  $(-\infty, b_1) \cup (b_2, \infty)$ ; 3)  $(-\infty, \infty)$ ; or 4)  $\emptyset$ , where  $\emptyset$  denotes the empty set. The CLR confidence regions could be any of the first three forms just given.

The LM confidence region could be any of 1)  $(b_1, b_2) \cup (b_3, b_4)$ ; 2)  $(-\infty, b_1) \cup (b_2, b_3) \cup (b_4, \infty)$ ; or 3)  $(-\infty, \infty)$ .

[Mikusheva \(2010\)](#) provides strong reasons for not using the LM test. While the CLR confidence regions are theoretically better than those for AR in overidentified models, they are not guaranteed to be shorter in a given application.

### 7.7.5 Minimum distance-based tests and confidence regions

[Stock and Wright \(2000\)](#) considered weak-instrument inference for optimal GMM estimators, including nonlinear GMM. They proposed so-called s tests and confidence intervals, where s denotes a suitable objective function, and considered homoskedastic and heteroskedastic errors. In the simplest case of linear simultaneous equations the s confidence intervals reduce to AR confidence intervals. [Kleibergen \(2005\)](#) proposed extensions of the CLR and LM tests to the continuous updating GMM estimator.

[Magnusson \(2010\)](#) proposed tests based on minimum distance (MD) estimation rather than GMM estimation. This bases estimation of the key structural parameter  $\beta_1$  on only the reduced-form parameter estimates. The advantage is that weak instruments pose no problem for estimating the first-stage model parameters. And various estimators can be used for the variance of reduced-form estimators, allowing for heteroskedastic–robust and cluster–robust standard errors; the theory is not limited to i.i.d. errors.

As an example, consider the model of section [7.7.2](#). Then the reduced form for  $y_1$  had the restriction  $\alpha = \Pi_2 \times \beta_1$ , which implies  $\alpha - \Pi_2 \times \beta_1 = \mathbf{0}$ . This suggests estimating the structural parameters  $\beta_1$  as the solution to  $\hat{\alpha} - \hat{\Pi}_2 \times \beta_1 = \mathbf{0}$ , where  $\hat{\alpha}$  and  $\hat{\Pi}_2$  are obtained from estimation of the reduced forms for, respectively,  $y_1$  and  $\mathbf{y}_2$ . In the just-identified case, there is a solution for  $\hat{\alpha}$ . In the overidentified case, we use the MD estimator, which minimizes a quadratic form in  $(\hat{\alpha} - \hat{\Pi}_2 \times \beta_1)$ .

More generally, let  $\beta$  denote a vector of structural form estimators we wish to estimate, let  $\pi$  denote a vector of reduced-form parameters, and

suppose that the model implies  $\mathbf{h}(\boldsymbol{\beta}, \boldsymbol{\pi}) = \mathbf{0}$ . Then the MD estimator minimizes the quadratic form

$$Q(\boldsymbol{\beta}) = \mathbf{h}(\boldsymbol{\beta}, \widehat{\boldsymbol{\pi}})' \widehat{\Lambda}^{-1} \mathbf{h}(\boldsymbol{\beta}, \widehat{\boldsymbol{\pi}}) \quad (7.17)$$

where  $\widehat{\Lambda}$  is a consistent estimate of  $\{\partial\mathbf{h}(\boldsymbol{\beta}, \boldsymbol{\pi})/\partial\boldsymbol{\pi}'\} \text{Var}(\widehat{\boldsymbol{\pi}}) \{\partial\mathbf{h}(\boldsymbol{\beta}, \boldsymbol{\pi})'/\partial\boldsymbol{\pi}\}$ . See, for example, [Cameron and Trivedi \(2005, 202\)](#) or [Wooldridge \(2010, 545\)](#).

[Magnusson \(2010\)](#) proposed MD analogs of the AR, CLR, and LM tests and confidence intervals.

#### The rivtest command

The MD method can be applied to both linear and nonlinear models. The community-contributed `rivtest` command ([Finlay and Magnusson 2009](#)) enables weak-instrument inference on a single endogenous regressor following regression using the `ivregress`, `ivreg2`, `ivprobit`, and `ivtobit` commands. The option `null()` allows specification of the null hypothesis; the default is  $\beta = 0$ . The option `ci` yields confidence intervals.

Continuing the current example, we obtain

```

. * rivtest: Weak instrument robust inference for non-i.i.d. errors
. qui ivregress 2sls ldrugexp (hi_empunion=ssiratio firmsz) $x2list, vce(robust)
. rivtest, ci null(0)
Estimating confidence sets over grid points
----- 1 ----- 2 ----- 3 ----- 4 ----- 5
..... 50
..... 100
Weak instrument robust tests and confidence sets for linear IV with robust VCE
H0: beta[ldrugexp:hi_empunion] = 0

```

Test	Statistic	p-value	95% Confidence Set
CLR	stat(.) = 26.36	Prob > stat = 0.0000	[-1.25987, -.537304]
AR	chi2(2) = 29.99	Prob > chi2 = 0.0000	[-1.12439, -.657732]
LM	chi2(1) = 25.97	Prob > chi2 = 0.0000	[-1.27492, -.537304]
J	chi2(1) = 4.02	Prob > chi2 = 0.0450	
LM-J	H0 rejected at 5% level		[-1.28998, -.522251]
Wald	chi2(1) = 21.97	Prob > chi2 = 0.0000	[-1.26363, -.518488]

Note: Wald test not robust to weak instruments. Confidence sets estimated for 100 points in [-1.63621, -0.145915].

The first three tests are heteroskedastic–robust versions of the weak-instrument CLR, AR, and LM tests. In overidentified models, the AR test that  $\alpha = \Pi_2(\beta - \beta_0) = 0$  decomposes into the sum of the LM test and the overidentification J test—here  $29.99 = 25.97 + 4.02$ . The LM-J test is a weighted combination of the two tests. The default is to put 80% on the LM test, so the LM test is performed at level 0.04, and the J test at level 0.01.

Unlike for the `condivreg` command, there are no disjoint confidence regions, though the grid search using program defaults is only over the range  $(-1.636, -0.146)$ .

### The `weakiv` command

The community-contributed command `weakiv` ([Finlay, Magnusson, and Schaffer 2014](#)) is an updated and expanded version of the `rivtest` command. It allows for multiple endogenous regressors, covers a wide range of robust variance–covariance estimators, including two-way clustering and HAC, and covers some panel commands.

The `weakiv` command can be implemented as a postestimation command for linear IV estimation after `ivregress`, `xtivreg`, `xtabond`, `ivtobit`, and

`ivprobit`, as well as after the community-contributed commands `ivreg2` and `xtivreg2`. Alternatively, it can be used as a stand-alone command, with the user providing the specification of the model.

We apply the `weakiv` command to the current example following `ivregress` with heteroskedastic-robust standard errors.

```
. * weakiv: Weak instrument robust inference for non-i.i.d. errors
. qui ivregress 2sls ldrugexp (hi_empunion=ssiratio firmsz) $x2list,
>      vce(robust)
. weakiv, null(0)
Estimating confidence sets over 100 grid points
+-----+
| 1 | 2 | 3 | 4 | 5 |
+-----+
..... 50
..... 100
```

Weak instrument robust tests and confidence sets for linear IV  
H0: beta[ldrugexp:hi\_empunion] = 0

Test	Statistic	p-value	Conf. level	Conf. Set
CLR	stat(.) = 23.26	0.0000	95%	[-1.25987, -.507198]
K	chi2(1) = 22.77	0.0000	95%	[-1.25987, -.507198]
J	chi2(1) = 4.50	0.0338	95% null set	
K-J	<n.a.>	0.0000	95% (96%, 99%)	[-1.28998, -.492144]
AR	chi2(2) = 27.28	0.0000	95%	[-1.12439, -.627625]
Wald	chi2(1) = 21.97	0.0000	95%	[-1.26363, -.518488]

Confidence sets estimated for 100 points in [-1.63621, -.145915].

Number of obs N = 10068.

Method = lagrange multiplier (LM). Weight on K in K-J test = 0.800.

Tests robust to heteroskedasticity.

Wald statistic in last row is based on `ivregress` estimation and is not robust to weak instruments.

The form of the output is similar to that for `rivtest`. The LM test is labeled the *K* test after [Kleibergen \(2005\)](#), who proposed the test. Surprisingly, there is some numerical difference in the test statistic values (aside from Wald) compared with `rivtest`.

The `rivtest` and `weakiv` commands implement versions of weak-instrument tests that can be robustified to non-i.i.d. errors. But they are not optimal when errors are not i.i.d., and there may be appreciable loss of test power, especially in models that have many excess instruments. Developing

more powerful tests when model errors are not i.i.d. is a current area of research.

## 7.8 Finite sample inference with weak instruments

The preceding weak-instrument inference relies on an alternative local-to-zero asymptotic theory. Nonetheless, this is still an asymptotic theory, and much of the theory is restricted to i.i.d. errors.

[Young \(Forthcoming\)](#) performs a Monte Carlo analysis of a sample of 30 leading applications of 2SLS estimators with a single endogenous regressor to evaluate the effect of non-i.i.d. errors and highly leveraged observations. Here the leverage (see section [3.6.3](#)) of the instruments for the  $i$ th independent observation is defined to equal the  $i$ th diagonal entry in  $\tilde{\mathbf{X}}_2(\tilde{\mathbf{X}}_2' \tilde{\mathbf{X}}_2)^{-1} \tilde{\mathbf{X}}_2$ , where  $\tilde{\mathbf{x}}_{2i}$  are the residuals from regressing the instruments  $\mathbf{x}_{2i}$  on the exogenous regressors  $\mathbf{x}_{1i}$ ; for clustered errors, a corresponding quantity is computed for the  $g$ th cluster. In the applications studied by [Young \(Forthcoming\)](#), some observations have very high leverage, not surprising because almost a half had less than 80 clusters or less than 80 observations if not clustered.

[Young \(Forthcoming\)](#) finds that using standard robust inference methods, the combination of non-i.i.d. errors and high leverage leads to first-stage  $F$  tests that greatly overstate the statistical significance of instruments (less so if the effective  $F$  statistic of [Montiel Olea and Pflueger \[2013\]](#) is used), tests on the coefficient of the endogenous regressor that greatly overstate statistical significance, and increase in the finite sample bias of 2SLS.

Simulations find that inference can be improved by using the jackknife or by performing either a pairs bootstrap or a wild bootstrap (see section [12.6](#)). Under strong instrument asymptotics, percentile- $t$  bootstraps provide an asymptotic refinement, while percentile bootstraps do not. Under weak-instrument asymptotics, however, none of these bootstraps provide an asymptotic refinement and in theory break down (as does the standard nonbootstrapped test) if in fact the model is unidentified. Here percentile bootstraps perform as well as percentile- $t$  bootstraps.

Section 12.6.5 implements a wild bootstrap of the Wald test for the 2SLS estimator using the community-contributed `boottest` command.

Additionally, it provides a percentile- $t$  wild bootstrap of the AR test, a bootstrap that does provide an asymptotic refinement because it relies only on strong-instrument asymptotic theory. [Young \(Forthcoming\)](#) did not consider the AR test, because the articles he studied did not use the AR test.

## 7.9 Other estimators

The literature suggests several alternative estimators that are asymptotically equivalent to 2SLS under strong instrument asymptotics, asymptotically different under weak-instrument asymptotics, and may have better finite-sample properties than 2SLS when instruments are weak.

We present some of these estimators, as well as the two-sample 2SLS estimator that enables estimation given data on the dependent variable from one sample and data on the endogenous regressor from a second sample, provided both samples have data on common instruments and exogenous regressors. Some additional estimators are available in the community-contributed `ivreg2` command ([Baum, Schaffer, and Stillman 2007](#)).

### 7.9.1 LIML estimator

The leading alternative to 2SLS is the LIML estimator. This estimator is based on the assumption of joint normality of errors in the structural and first-stage equations. It is an ML estimator for obvious reasons and is a limited-information estimator when compared with a full-information approach that specifies structural equations (rather than first-stage equations) for all endogenous variables in the model.

The LIML estimator preceded 2SLS but has been less widely used because it is known to be asymptotically equivalent to 2SLS. Both are special cases of the  $k$ -class estimators. The two estimators differ in finite samples, however, because of differences in the weights placed on instruments. Research has found that LIML has some desirable finite-sample properties, especially if the instruments are not strong. For example, several studies have shown that LIML has a smaller bias than either 2SLS or GMM.

The LIML estimator is a special case of the so-called  $k$ -class estimator, defined as

$$\widehat{\beta}_{k\text{-class}} = \left\{ \mathbf{X}'(\mathbf{I} - k\mathbf{M}_z)^{-1}\mathbf{X} \right\}^{-1} \mathbf{X}'(\mathbf{I} - k\mathbf{M}_z)^{-1}\mathbf{y}$$

where the structural equation is denoted here as  $\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$ . The LIML estimator sets  $k$  equal to the minimum eigenvalue of  $(\mathbf{Y}'\mathbf{M}_Z\mathbf{Y})^{-1/2}\mathbf{Y}'\mathbf{M}_{\mathbf{X}_1}\mathbf{Y}(\mathbf{Y}'\mathbf{M}_Z\mathbf{Y})^{-1/2}$ , where  $\mathbf{M}_{\mathbf{X}_1} = \mathbf{I} - \mathbf{X}_1(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1$ ,  $\mathbf{M}_Z = \mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}$ , and the first-stage equations are  $\mathbf{Y} = \mathbf{Z}\Pi + \mathbf{V}$ .

The estimator has a default VCE of

$$\widehat{V}(\widehat{\beta}_{k\text{-class}}) = s^2 \left\{ \mathbf{X}'(\mathbf{I} - k\mathbf{M}_Z)^{-1}\mathbf{X} \right\}^{-1}$$

where  $s^2 = \widehat{\mathbf{u}}'\widehat{\mathbf{u}}/N$  under the assumption that the errors  $\mathbf{u}$  and  $\mathbf{V}$  are homoskedastic. A leading  $k$ -class estimator is the 2SLS estimator, when  $k = 1$

The LIML estimator is obtained by using the `ivregress liml` command rather than `ivregress 2sls`. The `vce(robust)` option provides a robust estimate of the VCE for LIML when errors are heteroskedastic. In that case, the LIML estimator remains asymptotically equivalent to 2SLS. But in finite samples, studies suggest LIML may be better.

An alternative command that is equivalent to the `ivregress liml` command is `eregress`, introduced in section [7.4.9](#), which can be used if we assume that the model is linear and recursive (see section [7.2.3](#) for a definition of this concept). For example, in a two-equation model in which  $y_1$  depends upon a second endogenous variable  $y_2$  and an exogenous variable  $x$ , and an exogenous instrumental variable  $z$  is also available, the command `eregress y1 x, endogenous(y2 = x z)` produces output identical to that from executing the `ivregress liml` command.

The community-contributed `mivreg` command ([Anatolyev and Skolkova 2019](#)) considers a number of variations to LIML, including an adjustment due to [Fuller \(1977\)](#), that are more robust when there are many (possibly weak) instruments.

## 7.9.2 Jackknife IV estimator

The jackknife IV estimator (JIVE) eliminates the correlation between the first-stage fitted values and the structural-equation error term that is one source of bias of the traditional 2SLS estimator. The hope is that this may lead to smaller bias in the estimator.

Let the subscript  $(-i)$  denote the leave-one-out operation that drops the  $i$ th observation. Denote the structural equation by  $\mathbf{y}_i = \mathbf{x}'_i \boldsymbol{\beta} + \mathbf{u}_i$ , and consider first-stage equations for both endogenous and exogenous regressors, so  $\mathbf{x}'_i = \mathbf{z}'_i \boldsymbol{\Pi} + \mathbf{v}_i$ . Then, for each  $i = 1, \dots, N$ , we estimate the parameters of the first-stage model with the  $i$ th observation deleted, regressing  $\mathbf{X}_{(-i)}$  on  $\mathbf{Z}_{(-i)}$ , and, given estimate  $\widehat{\boldsymbol{\Pi}}_i$ , construct the instrument for observation  $i$  as  $\widetilde{\mathbf{x}}'_i = \mathbf{z}'_i \widehat{\boldsymbol{\Pi}}_i$ . Combining for  $i = 1, \dots, N$  yields an instrument matrix denoted by  $\widetilde{\mathbf{X}}_{(-i)}$  with the  $i$ th row  $\widetilde{\mathbf{x}}'_i$ , leading to the JIVE

$$\widetilde{\boldsymbol{\beta}}_{\text{JIVE}} = \left( \widetilde{\mathbf{X}}'_{(-i)} \mathbf{X} \right)^{-1} \widetilde{\mathbf{X}}'_{(-i)} \mathbf{y}$$

The community-contributed `jive` command ([Poi 2006](#)) has syntax that is similar to `ivregress`. The variants, specified as a command option, are `ujive1` and `ujive2` ([Angrist, Imbens, and Krueger 1999](#)) and `jive1` and `jive2` ([Blomquist and Dahlberg 1999](#)). The default is `ujive1`. The `robust` option gives heteroskedasticity-robust standard errors.

There is mixed evidence to date on the benefits of using JIVE; see the articles cited above and [Davidson and MacKinnon \(2006\)](#). Caution should be exercised in its use.

## 7.9.3 Comparison of 2SLS, LIML, JIVE, and GMM

We compare several estimators for an overidentified model with four instruments for `hi_empunion`. For the JIVE estimator, we use the community-contributed `jive` command, and for optimal GMM with heteroskedastic

errors, we use both `ivregress gmm` and the community-contributed `ivreg2` command ([Baum, Schaffer, and Stillman 2007](#)). We have

```
. * Variants of IV estimators: 2SLS, LIML, JIVE, GMM_het, GMM-het using IVREG2
. global ivmodel "ldrugexp $x2list
> (hi_empunion = ssiratio lowincome multlc firmsz)"
. qui ivregress 2sls $ivmodel, vce(robust)
. estimates store TWOALS
. qui ivregress liml $ivmodel, vce(robust)
. estimates store LIML
. qui jive $ivmodel, robust
. estimates store JIVE
. qui ivregress gmm $ivmodel, wmatrix(robust)
. estimates store GMM_het
. qui ivreg2 $ivmodel, gmm2s robust
. estimates store IVREG2
. estimates table TWOALS LIML JIVE GMM_het IVREG2, b(%7.4f) se
```

Variable	TWOALS	LIML	JIVE	GMM_het	IVREG2
hi_empunion	-0.8174 0.1706	-0.8612 0.1797	-0.8550 0.1783	-0.7809 0.1691	-0.7809 0.1699
totchr	0.4493 0.0100	0.4497 0.0101	0.4497 0.0101	0.4489 0.0100	0.4489 0.0100
age	-0.0123 0.0027	-0.0127 0.0027	-0.0127 0.0027	-0.0120 0.0026	-0.0120 0.0027
female	-0.0145 0.0303	-0.0180 0.0308	-0.0175 0.0307	-0.0087 0.0301	-0.0087 0.0302
blhisp	-0.2094 0.0381	-0.2123 0.0385	-0.2119 0.0384	-0.2015 0.0378	-0.2015 0.0380
linc	0.0820 0.0198	0.0855 0.0204	0.0850 0.0203	0.0783 0.0196	0.0783 0.0197
_cons	6.6982 0.2335	6.7401 0.2403	6.7341 0.2392	6.6695 0.2321	6.6695 0.2330

Legend: b/se

Here there is little variation across estimators in estimated coefficients and standard errors. As expected, the last two columns give exactly the same coefficient estimates, though the standard errors differ slightly.

#### 7.9.4 Two-sample 2SLS

Two-sample 2SLS is a variation of 2SLS that can be used when the complete data necessary for regular 2SLS are unavailable. As usual, we wish to fit the structural model  $y_{1i} = \beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + u_i$ , where the single endogenous regressor  $y_{2i}$  has first-stage equation  $y_{2i} = \mathbf{x}'_{1i} \boldsymbol{\pi}_1 + \mathbf{x}'_{2i} \boldsymbol{\pi}_2 + v_i$ .

The two-sample IV method enables consistent estimation when one dataset has data on  $y_{1i}$ ,  $\mathbf{x}_{1i}$ , and  $\mathbf{x}_{2i}$ , but not on the endogenous regressor  $y_{2i}$ , and a second dataset has data on  $y_{2i}$ ,  $\mathbf{x}_{1i}$ , and  $\mathbf{x}_{2i}$ , but not on  $y_{1i}$ , the dependent variable in the structural equation.

The two-sample 2SLS estimator is obtained as follows. First, OLS regression in the second sample is used to obtain estimates  $\hat{\boldsymbol{\pi}}_1$  and  $\hat{\boldsymbol{\pi}}_2$ . Second, these estimates are used to obtain prediction  $\hat{y}_{2i}$  using the first sample data. Third, using the first sample, we perform OLS regression of  $y_{1i}$  on  $\hat{y}_{2i}$  and  $\mathbf{x}_{1i}$ . This method uses the two-stage interpretation of the 2SLS estimator.

Inference needs to control for the final regression involving a generated regressor. Initial research did so for homoskedastic errors. [Pacini and Windmeijer \(2016\)](#) extend this to heteroskedastic errors, and the appendix to their article provides Stata code for implementation.

In general, a generated regressor such as  $\hat{y}_{2i}$  leads to attenuation bias. This becomes substantial for the two-sample 2SLS estimator when the instruments are weak and there are many instruments. [Choi, Gu, and Shen \(2018\)](#) provide weak-instrument inference in this case when errors are homoskedastic or heteroskedastic.

## 7.10 Three-stage least-squares systems estimation

The preceding estimators are asymmetric in that they specify a structural equation for only one variable, rather than for all endogenous variables. For example, we specified a structural model for `1drugexp` but not one for `hi_empunion`. A more complete model specifies structural equations for all endogenous variables.

Consider a multiequation model with  $m$  ( $\geq 2$ ) linear structural equations, each of the form

$$y_{ji} = \mathbf{y}'_{ji}\boldsymbol{\beta}_j + \mathbf{x}'_{ji}\boldsymbol{\beta}_{j2} + u_{ji}, \quad j = 1, \dots, m; \quad i = 1, \dots, N$$

For each of the  $m$  endogenous regressors  $y_j$ , we specify a structural equation with the endogenous regressors  $\mathbf{y}_j$ , the subset of endogenous variables that determine  $y_j$ , and the exogenous regressors  $\mathbf{x}_j$ , the subset of exogenous variables that determine  $y_j$ . Model identification is secured by rank and order conditions, given in standard graduate texts, requiring that some of the endogenous or exogenous regressors be excluded from each  $y_j$  equation.

The preceding IV estimators remain valid in this system. And specification of the full system can aid in providing instruments because any exogenous regressors in the system that do not appear in  $\mathbf{x}_j$  can be used as instruments for  $y_j$ .

Under the strong assumptions of error independence across  $i$  and that for the  $i$ th observation  $\text{Cov}(u_{ij}, u_{ik}) = \sigma_{ij}$ , more efficient estimation is possible by exploiting cross-equation correlation of errors, just as for the seemingly unrelated regressions model discussed in section 6.8. This estimator is called the three-stage least-squares (3SLS) estimator. Note that the assumptions include within-equation homoskedasticity and that the `reg3` command does not provide robust standard errors as an option.

For the example below, we need to provide a structural model for `hi_empunion` in addition to the structural model already specified for

`ldrugexp`. We suppose that `hi_empunion` depends on the single instrument `ssiratio`, on `ldrugexp`, and on `female` and `blhisp`. This means that we are (arbitrarily) excluding two regressors, `age` and `linc`. This ensures that the `hi_empunion` equation is overidentified. If instead it was just identified, then the system would be just identified because the `ldrugexp` is just identified, and 3SLS would reduce to equation-by-equation 2SLS.

The syntax for the `reg3` command is similar to that for `sureg`, with each equation specified in a separate set of parentheses. The endogenous variables in the system are simply determined because they are given as the first variable in each set of parentheses. We have

```
. * 3SLS estimation requires errors to be homoskedastic
. reg3 (ldrugexp hi_empunion totchr age female blhisp linc)
>      (hi_empunion ldrugexp totchr female blhisp ssiratio)
```

Three-stage least-squares regression

Equation	Obs	Params	RMSE	"R-squared"	chi2	P>chi2
ldrugexp	10,068	6	1.300469	0.0877	1955.02	0.0000
hi_empunion	10,068	5	1.6448	-10.4552	68.94	0.0000
<hr/>						
	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
ldrugexp						
hi_empunion	-.7890506	.1874758	-4.21	0.000	-1.156496	-.4216048
totchr	.4491568	.010287	43.66	0.000	.4289948	.4693189
age	-.0131018	.002546	-5.15	0.000	-.0180919	-.0081118
female	-.0126266	.0304786	-0.41	0.679	-.0723634	.0471103
blhisp	-.2114008	.0378127	-5.59	0.000	-.2855124	-.1372892
linc	.0724828	.0179903	4.03	0.000	.0372225	.1077431
_cons	6.7731	.2221739	30.49	0.000	6.337647	7.208553
hi_empunion						
ldrugexp	1.290877	.3169658	4.07	0.000	.6696355	1.912119
totchr	-.5529128	.1388413	-3.98	0.000	-.8250366	-.2807889
female	-.1292064	.0354309	-3.65	0.000	-.1986498	-.059763
blhisp	.150607	.0683069	2.20	0.027	.0167279	.2844861
ssiratio	-.4440772	.0594459	-7.47	0.000	-.560589	-.3275654
_cons	-6.668402	1.78005	-3.75	0.000	-10.15724	-3.179569

Endogenous variables: `ldrugexp hi_empunion`

Exogenous variables: `totchr age female blhisp linc ssiratio`

## 7.11 Additional resources

The `ivregress` command is the key command for linear IV models. The community-contributed `ivreg2` command ([Baum, Schaffer, and Stillman 2007](#)) has many additional features, including two-way cluster-robust standard errors and additional tests of endogeneity. The approach generalizes to nonlinear 2SLS and GMM; an example is the `ivpoisson gmm` command.

A major concern is that asymptotic theory performs poorly when instruments are weak or there are many instruments, an area that is still one of active research. One approach uses alternative inferential methods, the simplest of which is based on the Anderson–Rubin test. For a just-identified single endogenous regressor, one need only use the `AR` test. Otherwise, a range of methods, including the `AR` test, have been proposed. The community-contributed `condivreg` command ([Mikusheva and Poi 2006](#)) enables inference with weak instruments assuming i.i.d. errors. The community-contributed command `weakiv` ([Finlay and Magnusson 2009](#)) extends the scope of the `condivreg` command by allowing non-i.i.d. errors. An alternative approach uses alternative estimators to 2SLS or LIML. The community-contributed `jive` command ([Poi 2006](#)) performs JIVE estimation. The community-contributed `mivreg` command ([Anatolyev and Skolkova 2019](#)) considers a number of variations to LIML. For many instruments, machine-learning regularization methods such as the lasso can be used; see section 28.8.

For many nonlinear models with endogenous regressors, attention is restricted to ML estimation of models with a recursive structure and normal errors. The extended regression model commands presented in sections 23.7 and 25.3 fit these models, and special cases of these commands overlap with, for example, the `ivprobit` and `ivtobit` commands. The related endogenous treatment commands `etregress`, `eteffects`, and `etpoisson` presented in section 25.4 explicitly control for a binary endogenous regressor.

## 7.12 Exercises

1. Estimate by 2SLS the same regression model as in section [7.4.4](#), with the instruments `multlc` and `firmsz`. Compare the 2SLS estimates with OLS estimates. Perform a test of endogeneity of `hi_empunion`. Perform a test of overidentification. State what you conclude. Throughout this exercise, perform inference that is robust to heteroskedasticity.
2. Repeat exercise 1 using optimal GMM.
3. Use the model and instruments of exercise 1. Compare the following estimators: 2SLS, LIML, and optimal GMM given heteroskedastic errors. For the last model, estimate the parameters by using the community-contributed `ivreg2` command in addition to `ivregress`.
4. Use the model of exercise 1. Compare 2SLS estimates as the instruments `ssiratio`, `lowincome`, `multlc`, and `firmsz` are progressively added.
5. Use the model and instruments of exercise 1. Perform appropriate diagnostics and tests for weak instruments using the 2SLS estimator. State what you conclude. Throughout this exercise, perform inference assuming errors are i.i.d.
6. Use the model and instruments of exercise 1. Use the community-contributed `condivreg` command to perform inference for the 2SLS estimator. Compare the results with those using conventional asymptotics.
7. Use the model and instruments of exercise 1. Use the community-contributed `jive` command, and compare estimates and standard errors from the four different variants of JIVE and from optimal GMM. Throughout this exercise, perform inference that is robust to heteroskedasticity.
8. Fit the 3SLS model of section [7.10](#), and compare the 3SLS coefficient estimates and standard errors in the `1drugexp` equation with those from 2SLS estimation (with default standard errors).
9. This question considers the same earnings–schooling dataset as that analyzed in [Cameron and Trivedi \(2005, 111\)](#). The data are in `mus207klingdata.dta`. The `describe` command provides descriptions of the regressors. There are three endogenous regressors (years of schooling, years of work experience, and experience-squared) and three instruments (a college proximity indicator, age, and age-squared). Interest lies in the coefficient of schooling. Perform appropriate diagnostics and tests for weak instruments for the following model. State what you conclude. The following commands yield the IV estimator:

```

. use mus207klingdata.dta, clear
. global x2list black south76 smsa76 reg2-reg9 smsa66
> sinmom14 nodaded nomomed daded momed famed1-famed8
. ivregress 2sls wage76 (grade76 exp76 expsq76 = col4 age76 agesq76) $x2list,
> vce(robust) perfect
. estat firststage

```

10. Use the same dataset as the previous question. Treat only `grade76` as endogenous, let `exp76` and `expsq76` be exogenous, and use `col4` as the only instrument. Perform appropriate diagnostics and tests for a weak instrument, and state what you conclude. Then, use the community-contributed `condivreg` command to perform inference, and compare the results with those using conventional asymptotics.
11. When an endogenous variable enters the regression nonlinearly, the obvious IV estimator is inconsistent and a modification is needed. Specifically, suppose  $y_1 = \beta y_2^2 + u$  and the first-stage equation for  $y_2$  is  $y_2 = \pi_2 z + v$ , where the zero-mean errors  $u$  and  $v$  are correlated. Here the endogenous regressor appears in the structural equation as  $y_2^2$  rather than  $y_2$ . The IV estimator is  $\hat{\beta}_{IV} = (\sum_i z_i y_{2i}^2)^{-1} \sum_i z_i y_{1i}$ . This can be implemented by a regular IV regression of  $y_1$  on  $y_2^2$  with the instrument  $z$ : regress  $y_2^2$  on  $z$  and then regress  $y_1$  on the first-stage prediction  $\hat{y}_2^2$ . If instead we regress  $y_2$  on  $z$  at the first stage, giving  $\hat{y}_2$ , and then regress  $y_1$  on  $(\hat{y}_2)^2$ , an inconsistent estimate is obtained. Generate a simulation sample to demonstrate these points. Consider whether this example can be generalized to other nonlinear models where the nonlinearity is in regressors only, so that  $y_1 = g(y_2)'\beta + u$ , where  $g(y_2)$  is a nonlinear function of  $y_2$ .
12. This exercise is based on artificially generated data. Set sample size to 1,000. Set seed to 10101. Generate a sample using the following DGP:  
 $u \sim N(0, 1); e \sim 0.6u + N(0, 1); z \sim \text{uniform}(0, 1);$   
 $y_2 = z + 0.5z^2 + 0.0625z^3 + e; x \sim N(0, 9); y_1 = y_2 + 0.5y_2^2 + x + u.$   
Estimate the  $y_1$  equation by OLS. Theoretically, the estimates of the equation are inconsistent. Compare all the slope coefficients with their true values, and discuss whether the direction of the bias is as you might expect. OLS is inconsistent in this case, yet some coefficients are estimated close to their true values; explain why. Run the same regression again with the intercept constrained to zero, and comment on any differences that you might observe.
13. Continuing the previous exercise, estimate three variants of the  $y_1$ -equation by 2SLS using as instruments i)  $x, z$ , and  $z^2$ ; ii)  $x, z, z^2$ , and  $z^3$ ; iii)  $\hat{y}_2$  and  $\hat{y}_2^2$ , where  $\hat{y}_2$  is the fitted value of  $y_2$  from the reduced form of  $y_2$ . Are the instruments chosen in each case valid and relevant? Which of the three estimates would you prefer on theoretical grounds? Explain your answer.

With reference to the 2SLS estimates used earlier, apply the test of overidentification where appropriate, and interpret your results.

14. This question continues with the model presented in question 12 above. Modify the specification of the  $y_1$  equation such that the equation errors are heteroskedastic, with skedasticity being a quadratic function of  $x$ ; that is,  $\text{Var}(u) = h(x) \times N(0, 1)$ . Reestimate the  $y_1$  equation by 2SLS and two-step GMM using as instruments  $x, z, z^2, z^3$ . In both cases, apply the test of overidentifying restrictions. Compare the results, and comment on the outcome of the test.



# **Chapter 8**

## **Linear panel-data models: Basics**

## 8.1 Introduction

Panel data or longitudinal data are repeated measurements at different points in time on the same individual unit, such as person, firm, state, or country. Regressions can then capture both variation over units, similar to regression on cross-sectional data, and variation over time.

Panel-data methods are more complicated than cross-sectional–data methods. The standard errors of panel-data estimators need to be adjusted because each additional time period of data in general is not independent of previous periods. Panel data requires the use of much richer models and estimation methods. Also, different areas of applied statistics use different methods for essentially the same data. The Stata `xt` commands, where `xt` is an acronym for cross-sectional time series, cover many of these methods.

We focus on methods for a short panel, meaning data on many individual units and few time periods. Examples include longitudinal surveys of many individuals and panel datasets on many firms. And we emphasize microeconometrics methods that attempt to estimate key marginal effects that can be given a causative interpretation.

The essential panel-data methods are given in this chapter, most notably, the important distinction between fixed-effects (FE) and random-effects (RE) models. The panel methods overlap considerably with those presented in sections [6.4–6.7](#).

Chapter [9](#) presents many other panel-data methods for the linear model, including those for instrumental-variables (IV) estimation, estimation when lagged dependent variables are regressors, estimation when panels are long rather than short, and estimation of mixed models with slope parameters that vary across individuals. Nonlinear panel models are presented in chapter 22.

## 8.2 Panel-data methods overview

There are many types of panel data and goals of panel-data analysis, leading to different models and estimators for panel data. We provide an overview in this section, with subsequent sections illustrating many of the various models and estimation methods.

### 8.2.1 Some basic considerations

First, panel data are usually observed at regular time intervals, as is the case for most time-series data. A common exception is growth curve analysis where, for example, children are observed at several irregularly spaced intervals in time and a measure such as height or IQ is regressed on a polynomial in age.

Second, panel data can be balanced, meaning all individual units are observed in all time periods ( $T_i = T$  for all  $i$ ), or unbalanced ( $T_i \neq T$  for some  $i$ ). Most `xt` commands can be applied to both balanced and unbalanced data. In either case, however, estimator consistency requires that the sample-selection process not lead to errors being correlated with regressors. Loosely speaking, the missingness is for random reasons rather than systematic reasons; see section 19.10.

Third, the dataset may be a short panel (few time periods and many individuals); a long panel (many time periods and few individuals); or both (many time periods and many individuals). This distinction has consequences for both estimation and inference.

Fourth, model errors are very likely correlated. Microeconometrics methods emphasize correlation (or clustering) over time for a given individual, with independence over individual units. For some panel datasets, such as country panels, there additionally may be correlation across individuals. Regardless of the assumptions made, some correction to default ordinary least-squares (OLS) standard errors is usually necessary, and efficiency gains using generalized least squares (GLS) may be possible.

Fifth, regression coefficient identification for some estimators can depend on regressor type. Some regressors, such as gender, may be time invariant with  $x_{it} = x_i$  for all  $t$ . Some regressors, such as an overall time trend, may be individual invariant with  $x_{it} = x_t$  for all  $i$ . And some may vary over both time and individuals.

Sixth, some or all model coefficients may vary across individuals or over time.

Seventh, the microeconomics literature emphasizes the FE model. This model, explained in the next section, permits regressors to be endogenous provided that they are correlated only with a time-invariant component of the error. Most other branches of applied statistics instead emphasize the RE model, which assumes that regressors are completely exogenous.

Finally, panel data permit estimation of dynamic models where lagged dependent variables may be regressors. Most panel-data analyses use models without this complication.

In this chapter, we focus on short panels ( $T$  fixed and  $N \rightarrow \infty$ ) with model errors assumed to be independent over individuals. We consider linear models with and without fixed effects—static models in this chapter and dynamic models in the subsequent chapter. Long panels are treated separately in section 9.5.

Most applications in this chapter use balanced panels. Unbalanced panels arise from missing data often arising from panel attrition, which simply means that respondents drop out of the panel altogether or have gaps in their participation in the survey. Most commands can also be applied to unbalanced panels. However, panel attrition can lead to inconsistent parameter estimates if it is not random after controlling for observable variables. Methods that correct for any bias that arises because of panel attrition are presented in section 19.11.

### 8.2.2 Some basic panel models

There are several different linear models for panel data.

The fundamental distinction is that between FE and RE models. The term “fixed effects” is misleading because in both types of models, individual-level effects are random. FE models have the added complication that regressors may be correlated with the individual-level effects so that consistent estimation of regression parameters requires eliminating or controlling for the fixed effects.

### **Individual-effects model**

The individual-specific effects model for the scalar dependent variable  $y_{it}$  specifies that

$$y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + \varepsilon_{it}, \quad t = 1, \dots, T_i, \quad i = 1, \dots, N \quad (8.1)$$

where  $\mathbf{x}_{it}$  are regressors,  $\alpha_i$  are random individual-specific effects, and  $\varepsilon_{it}$  is an idiosyncratic error. For simplicity, we mostly present results with  $T_i = T$ .

Two quite different models for the  $\alpha_i$  are the FE and RE models.

### **Fixed-effects model**

In the FE model, the  $\alpha_i$  in (8.1) are permitted to be correlated with the regressors  $\mathbf{x}_{it}$ . This allows a limited form of endogeneity. We view the error in (8.1) as  $u_{it} = \alpha_i + \varepsilon_{it}$  and permit  $\mathbf{x}_{it}$  to be correlated with the time-invariant component of the error ( $\alpha_i$ ), while continuing to assume that  $\mathbf{x}_{it}$  is uncorrelated with the idiosyncratic error  $\varepsilon_{it}$ . For example, we assume that if regressors in an earnings regression are correlated with unobserved ability, they are correlated only with the time-invariant component of ability, captured by  $\alpha_i$ .

One possible estimation method is to jointly estimate  $\alpha_1, \dots, \alpha_N$  and  $\beta$ . But for a short panel, asymptotic theory relies on  $N \rightarrow \infty$ , and here as  $N \rightarrow \infty$  so too does the number of fixed effects to estimate. This problem is called the incidental-parameters problem. Interest lies in estimating  $\beta$ , but first we need to control for the nuisance or incidental parameters,  $\alpha_i$ .

Instead, we can still consistently estimate  $\beta$ , for time-varying regressors, by appropriate differencing transformations detailed in sections [8.5](#) and [8.9](#) that eliminate  $\alpha_i$ .

The FE model implies that  $E(y_{it}|\alpha_i, \mathbf{x}_{it}) = \alpha_i + \mathbf{x}'_{it}\beta$ , assuming  $E(\varepsilon_{it}|\alpha_i, \mathbf{x}_{it}) = 0$ , so  $\beta_j = \partial E(y_{it}|\alpha_i, \mathbf{x}_{it})/\partial x_{j,it}$ . The attraction of the FE model is that we can obtain a consistent estimate of the marginal effect of the  $j$ th regressor on  $E(y_{it}|\alpha_i, \mathbf{x}_{it})$ , provided  $x_{j,it}$  is time varying, even if the regressors are endogenous (albeit, a limited form of endogeneity).

At the same time, knowledge of  $\beta$  does not give complete information on the process generating  $y_{it}$ . In particular for prediction, we need an estimate of  $E(y_{it}|\mathbf{x}_{it}) = E(\alpha_i|\mathbf{x}_{it}) + \mathbf{x}'_{it}\beta$ , and  $E(\alpha_i|\mathbf{x}_{it})$  cannot be consistently estimated in short panels.

In nonlinear FE models, these results need to be tempered. We cannot always eliminate  $\alpha_i$ , which is shown in section 22.2. And even if it is, consistent estimation of  $\beta$  may still not lead to a consistent estimate of the marginal effect  $\partial E(y_{it}|\alpha_i, \mathbf{x}_{it})/\partial x_{j,it}$ .

### **Random-effects model**

In the RE model, it is assumed that  $\alpha_i$  in [\(8.1\)](#) is purely random, a stronger assumption implying that  $\alpha_i$  is uncorrelated with the regressors.

Estimation is then by a feasible generalized least-squares (FGLS) estimator, given in section [8.6](#). Advantages of the RE model are that it yields estimates of all coefficients and hence marginal effects, even those of time-invariant regressors, and that  $E(y_{it}|\mathbf{x}_{it})$  can be estimated. The big disadvantage is that these estimates are inconsistent if the FE model is appropriate.

### **Correlated RE model**

A variation of the RE model controls for fixed effects by adding individual-specific means of time-varying regressors as additional regressors; see section [8.7.4](#).

### Pooled model or population-averaged model

Pooled models assume that regressors are exogenous and simply write the error as  $u_{it}$  rather than using the decomposition  $\alpha_i + \varepsilon_{it}$ . Then,

$$y_{it} = \alpha + \mathbf{x}'_{it}\beta + u_{it} \quad (8.2)$$

Note that  $\mathbf{x}_{it}$  here does not include a constant, whereas in cross-sectional chapters,  $\mathbf{x}_i$  additionally included a constant term.

OLS estimation of the parameters of this model is straightforward, but inference needs to control for likely correlation of the error  $u_{it}$  over time for a given individual (within correlation) and possible correlation over individuals (between correlation). FGLS estimation of (8.2) given an assumed model for the within correlation of  $u_{it}$  is presented in section 8.4. In the statistics literature, this is called a population-averaged (PA) model. Like RE estimators, consistency of the estimators requires that regressors be uncorrelated with  $u_{it}$ .

### Two-way effects model

A standard extension of the individual effects is a two-way effects model that allows the intercept to vary over individuals and over time:

$$y_{it} = \alpha_i + \gamma_t + \mathbf{x}'_{it}\beta + \varepsilon_{it} \quad (8.3)$$

For short panels, it is common to let the time effects  $\gamma_t$  be fixed effects. Then (8.3) reduces to (8.1), if the regressors in (8.1) include a set of time dummies (with one time dummy dropped to avoid the dummy-variable trap).

### Mixed linear models

If the RE model is appropriate, richer models can permit slope parameters to also vary over individuals or time. The mixed linear model (see section 6.7) is a hierarchical linear model that is quite flexible and permits random

parameter variation to depend on observable variables. The random-coefficients model is a special case that specifies

$$y_{it} = \alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta}_i + \varepsilon_{it}$$

where  $(\alpha_i \ \boldsymbol{\beta}'_i)' \sim (\boldsymbol{\beta}, \boldsymbol{\Sigma})$ . For a long panel with few individuals,  $\alpha_i$  and  $\boldsymbol{\beta}_i$  can instead be parameters that can be estimated by running separate regressions for each individual.

### 8.2.3 Cluster–robust inference

Various estimators for the preceding models are given in subsequent sections. These estimators are usually based on the assumption that the idiosyncratic error  $\varepsilon_{it} \sim (0, \sigma_\varepsilon^2)$ . This assumption is often not satisfied in panel applications. Then many panel estimators still retain consistency, provided that  $\varepsilon_{it}$  are independent over  $i$ , but reported standard errors are incorrect.

For short panels, we can obtain cluster–robust standard errors under the weaker assumptions that errors are independent across individuals and that  $N \rightarrow \infty$ . Specifically,  $E(\varepsilon_{it}\varepsilon_{js}) = 0$  for  $i \neq j$ ,  $E(\varepsilon_{it}\varepsilon_{is})$  is unrestricted, and  $\varepsilon_{it}$  may be heteroskedastic.

Where applicable, we use cluster–robust standard errors rather than the Stata defaults. In particular, while the inclusion of random or fixed effects  $\alpha_i$  can partly account for within-individual error correlation over time, in practice it is often insufficient, and failure to additionally cluster on the individual typically leads to underestimation of standard errors and overstatement of estimator precision.

For most, but not all, `xt` commands, the `vce(robust)` option is available. When this option is available, it produces cluster–robust standard errors, rather than heteroskedastic-robust standard errors, with clustering on the individual unit that is defined in the `xtset` command. In some applications, one should cluster at a higher level than the individual. For example, with individual-level panel data and the key regressor a policy variable that varies

at the region level, such as state, one should use the `vce(cluster clustvar)` option of the `xt` command, if it is available. Care is needed in using cluster-robust standard errors for panel estimators because in some cases, including some `xt` commands that have a `vce(robust)` option, consistent estimation may require that there be no within-individual error correlation. In some cases the `vce(bootstrap)` or `vce(jackknife)` option can be used to obtain cluster-robust standard errors because, for `xt` commands, these usually resample over clusters. But again, within-cluster correlation may lead to the more serious problem of inconsistent parameter estimation.

In a seminal article, [Bertrand, Duflo, and Mullainathan \(2004\)](#) analyzed state-year panel data with a state-level policy variable. They had two major findings. First, even if individual-level data are available, clustering should be on state, rather than on state-year pair. Second, when the number of states is few and clustering is on states, then the standard cluster-robust Wald test overrejects substantially.

Sections [6.2.4](#) and [6.4](#) provide considerable detail on cluster-robust inference for OLS, and [Cameron and Miller \(2015\)](#), [MacKinnon and Webb \(2020\)](#), and [MacKinnon, Nielsen, and Webb \(Forthcoming\)](#) provide surveys. The problem of inference with few clusters is detailed in section [6.4.6](#). A general method for inference with few clusters applies the percentile- $t$  method to a particular bootstrap, the wild cluster bootstrap; see section [12.6](#). The community-contributed `boottest` command ([Roodman et al. 2019](#)) presented in section [12.6.2](#) implements this bootstrap following a wide range of estimation commands, including the commands `regress`, `areg`, and `xtreg, fe` used in linear model analysis of panel data.

## 8.2.4 The `xtreg` command

The key command for estimation of the parameters of a linear panel-data model is the `xtreg` command, also used for clustered cross-sectional data; see chapter [6](#). The command syntax is

```
xtreg depvar [indepvars] [if] [in] [weight] [, options]
```

The individual identifier must first be declared with the `xtset` command.

The key model options are PA model (`pa`), FE model (`fe`), RE model (`re` and `mle`), and between-effects model (`be`). The individual models are discussed in detail in subsequent sections. The *weight* modifier is available only for `fe`, `mle`, and `pa`.

The `vce(robust)` option provides cluster-robust estimates of the standard errors for all models but `be`. Stata labels the estimated VCE as simply “Robust” because the use of `xtreg` implies that we are in a clustered setting.

### 8.2.5 Stata linear panel-data commands

Table 8.1 summarizes `xt` commands for viewing panel data and estimating the parameters of linear panel-data models.

**Table 8.1.** Summary of `xt` commands for linear panel models

Data summary	<code>xtset; xtdescribe; xtsum; xtdata; xtline; xttab; xttrans</code>
Pooled OLS	<code>regress</code>
Pooled FGLS	<code>xtgee, family(gaussian); xtgls; xtpcse</code>
RE	<code>xtreg, re; xtregar, re</code>
FE	<code>xtreg, fe; xtregar, fe</code>
Random slopes	<code>mixed; xtrc</code>
First-differences	<code>regress</code> (with differenced data)
Differences in differences	<code>xtdidregress</code>
Static IV	<code>xtivreg; xtaylor</code>
Dynamic IV	<code>xtabond; xtdpdsys; xtdpd</code>
Unit-root tests	<code>xtunitroot</code>
Cointegration tests	<code>xtcointtest</code>

The core methods for short panels, notably the data summary commands and the `xtreg` and `xtgee` commands, are presented in this chapter, with more specialized commands for panel IV estimation and for long panels presented

in chapter 9. Readers with long panels should look at section 9.5 (`xtgls`, `xtpcse`, `xtregar`), and data input may require first reading section 8.10. Some additional panel commands for censored regression and for nonlinear models are given in table 13.1.

## 8.3 Summary of panel data

In this section, we present various ways to summarize and view panel data and estimate a pooled OLS regression. The dataset used is a panel on log hourly wages and other variables for 595 people over the seven years 1976–1982.

### 8.3.1 Data description and summary statistics

The data, from [Baltagi and Khanti-Akom \(1990\)](#), were drawn from the Panel Study of Income Dynamics and are a corrected version of data originally used by [Cornwell and Rupert \(1988\)](#).

The dataset has the following data:

. * Read in dataset and describe				
. qui use mus208psid				
. describe				
Contains data from mus208psid.dta				
Observations:	4,165			A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e 1 Sep 2020 16:38 (_dta has notes)
Variables:	22			
Variable name	Storage type	Display format	Value label	Variable label
exp	float	%9.0g		Years of full-time work experience
wks	float	%9.0g		Weeks worked
occ	float	%9.0g		Occupation; occ==1 if in a blue-collar occupation
ind	float	%9.0g		Industry; ind==1 if working in a manufacturing industry
south	float	%9.0g		Residence; south==1 if in the South area
smsa	float	%9.0g		smsa==1 if in the standard metropolitan statistical area
ms	float	%9.0g		Marital status
fem	float	%9.0g		Female or male
union	float	%9.0g		If wage set be a union contract
ed	float	%9.0g		Years of education
blk	float	%9.0g		Black
lwage	float	%9.0g		log wage
id	float	%9.0g		Person ID
t	float	%9.0g		Year index
tdum1	byte	%8.0g		t==1.0000
tdum2	byte	%8.0g		t==2.0000
tdum3	byte	%8.0g		t==3.0000
tdum4	byte	%8.0g		t==4.0000
tdum5	byte	%8.0g		t==5.0000
tdum6	byte	%8.0g		t==6.0000
tdum7	byte	%8.0g		t==7.0000
exp2	float	%9.0g		Years of experience squared

Sorted by: id t

There are 4,165 individual–year pair observations. The variable labels describe the variables fairly clearly, though note that `lwage` is the log of hourly wage in cents, the indicator `fem` is 1 if female, `id` is the individual identifier, `t` is the year, and `exp2` is the square of `exp`.

Descriptive statistics can be obtained by using the command `summarize`:

```
. * Summary of dataset
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
exp	4,165	19.85378	10.96637	1	51
wks	4,165	46.81152	5.129098	5	52
occ	4,165	.5111645	.4999354	0	1
ind	4,165	.3954382	.4890033	0	1
south	4,165	.2902761	.4539442	0	1
smsa	4,165	.6537815	.475821	0	1
ms	4,165	.8144058	.3888256	0	1
fem	4,165	.112605	.3161473	0	1
union	4,165	.3639856	.4812023	0	1
ed	4,165	12.84538	2.787995	4	17
blk	4,165	.0722689	.2589637	0	1
lwage	4,165	6.676346	.4615122	4.60517	8.537
id	4,165	298	171.7821	1	595
t	4,165	4	2.00024	1	7
tdum1	4,165	.1428571	.3499691	0	1
tdum2	4,165	.1428571	.3499691	0	1
tdum3	4,165	.1428571	.3499691	0	1
tdum4	4,165	.1428571	.3499691	0	1
tdum5	4,165	.1428571	.3499691	0	1
tdum6	4,165	.1428571	.3499691	0	1
tdum7	4,165	.1428571	.3499691	0	1
exp2	4,165	514.405	496.9962	1	2601

The variables take on values that are within the expected ranges, and there are no missing values. Both men and women are included, though from the mean of `fem`, only 11% of the sample is female. Wages data are nonmissing in all years, and weeks worked are always positive, so the sample is restricted to individuals who work in all seven years.

### 8.3.2 Panel-data organization

The `xt` commands require that panel data be organized in so-called long form, with each observation a distinct individual–time pair, here an individual–year pair. Data may instead be organized in wide form, with a single observation combining data from all years for a given individual or combining data on all individuals for a given year. Then, the data need to be

converted from wide form to long form by using the `reshape` command presented in section [8.10](#).

Data organization can often be clear from listing the first few observations. For brevity, we list the first three observations for a few variables:

```
. * Organization of dataset
. list id t exp wks occ in 1/3, clean
      id    t    exp    wks    occ
1.    1    1      3     32      0
2.    1    2      4     43      0
3.    1    3      5     40      0
```

The first observation is for individual 1 in year 1, the second observation is for individual 1 in year 2, and so on. These data are thus in long form. From `summarize`, the panel identifier `id` takes on the values 1–595, and the time variable `t` takes on the values 1–7. In general, the panel identifier need just be a unique identifier, and the time variable could take on values of, for example, 76–82.

The panel-data `xt` commands require that, at a minimum, the panel identifier be declared. Many `xt` commands also require that the time identifier be declared. This is done by using the `xtset` command. Here we declare both identifiers:

```
. * Declare individual identifier and time identifier
. xtset id t
Panel variable: id (strongly balanced)
Time variable: t, 1 to 7
Delta: 1 unit
```

The panel identifier is given first, followed by the optional time identifier. The output indicates that data are available for all individuals in all time periods (strongly balanced), and the time variable increments uniformly by one.

When a Stata dataset is saved, the current settings, if any, from `xtset` are also saved. In this particular case, the original dataset `mus208psid.dta` already contained this information, so the preceding `xtset` command was

actually unnecessary. The `xtset` command without any arguments reveals the current settings, if any.

### 8.3.3 Panel-data description

Once the panel data are `xtset`, the `xtdescribe` command provides information about the extent to which the panel is unbalanced.

```
. * Panel description of dataset
. xtdescribe

    id: 1, 2, ..., 595                               n =      595
    t: 1, 2, ..., 7                                 T =       7
    Delta(t) = 1 unit
    Span(t) = 7 periods
    (id*t uniquely identifies each observation)

Distribution of T_i:   min      5%     25%     50%     75%     95%     max
                      7        7        7        7        7        7        7
    Freq.  Percent   Cum. |  Pattern
    595    100.00 100.00 |  1111111
    595    100.00          |  XXXXXXXX
```

In this case, all 595 individuals have exactly 7 years of data. The data are therefore balanced because, additionally, the earlier `summarize` command showed that there are no missing values. Section 22.3 provides an example of `xtdescribe` with unbalanced data and covers ways of balancing an unbalanced sample if so desired.

### 8.3.4 Within and between variation

Dependent variables and regressors can potentially vary over both time and individuals. Variation over time or a given individual is called within variation, and variation across individuals is called between variation. This distinction is important because estimators differ in their use of within and between variation. In particular, in the FE model, the coefficient of a regressor with little within variation will be imprecisely estimated and will be not identified if there is no within variation at all.

The `xtsum`, `xtdescribe`, and `xttrans` commands provide information on the relative importance of within variation and between variation of a variable.

We begin with `xsum`. The total variation (around grand mean  $\bar{x} = 1/NT \sum_i \sum_t x_{it}$ ) can be decomposed into within variation over time for each individual (around individual mean  $\bar{x}_i = 1/T \sum_t x_{it}$ ) and between variation across individuals (for  $\bar{x}$  around  $\bar{x}_i$ ). The corresponding decomposition for the variance is

$$\text{Within variance: } s_W^2 = \frac{1}{NT-1} \sum_i \sum_t (x_{it} - \bar{x}_i)^2 = \frac{1}{NT-1} \sum_i \sum_t (x_{it} - \bar{x}_i + \bar{x})^2$$

$$\text{Between variance: } s_B^2 = \frac{1}{N-1} \sum_i (\bar{x}_i - \bar{x})^2$$

$$\text{Overall variance: } s_O^2 = \frac{1}{NT-1} \sum_i \sum_t (x_{it} - \bar{x})^2$$

The second expression for  $s_W^2$  is equivalent to the first, because adding a constant does not change the variance, and is used at times because  $x_{it} - \bar{x}_i + \bar{x}$  is centered on  $\bar{x}$ , providing a sense of scale, whereas  $x_{it} - \bar{x}_i$  is centered on zero. For unbalanced data, replace  $NT$  in the formulas with  $\sum_i T_i$ . It can be shown that  $s_O^2 \simeq s_W^2 + s_B^2$ .

The `xsum` command provides this variance decomposition. We do this for selected regressors and obtain

```
. * Panel summary statistics: Within and between variation
. xtsum id t lwage ed exp exp2 wks south tdum1
```

Variable		Mean	Std. dev.	Min	Max	Observations
id	overall	298	171.7821	1	595	N = 4165
	between		171.906	1	595	n = 595
	within		0	298	298	T = 7
t	overall	4	2.00024	1	7	N = 4165
	between		0	4	4	n = 595
	within		2.00024	1	7	T = 7
lwage	overall	6.676346	.4615122	4.60517	8.537	N = 4165
	between		.3942387	5.3364	7.813596	n = 595
	within		.2404023	4.781808	8.621092	T = 7
ed	overall	12.84538	2.787995	4	17	N = 4165
	between		2.790006	4	17	n = 595
	within		0	12.84538	12.84538	T = 7
exp	overall	19.85378	10.96637	1	51	N = 4165
	between		10.79018	4	48	n = 595
	within		2.00024	16.85378	22.85378	T = 7
exp2	overall	514.405	496.9962	1	2601	N = 4165
	between		489.0495	20	2308	n = 595
	within		90.44581	231.405	807.405	T = 7
wks	overall	46.81152	5.129098	5	52	N = 4165
	between		3.284016	31.57143	51.57143	n = 595
	within		3.941881	12.2401	63.66867	T = 7
south	overall	.2902761	.4539442	0	1	N = 4165
	between		.4489462	0	1	n = 595
	within		.0693042	-.5668667	1.147419	T = 7
tdum1	overall	.1428571	.3499691	0	1	N = 4165
	between		0	.1428571	.1428571	n = 595
	within		.3499691	0	1	T = 7

Time-invariant regressors have zero within variation, so the individual identifier `id` and the variable `ed` are time invariant. Individual-invariant regressors have zero between variation, so the time identifier `t` and the time dummy `tdum1` are individual invariant. For all other variables but `wks`, there is more variation across individuals (between variation) than over time (within variation), so within estimation may lead to considerable efficiency loss. What is not clear from the output from `xtsum` is that while variable `exp` has nonzero within variation, it evolves deterministically because for this

sample,  $\exp$  increments by one with each additional period. The  $\min$  and  $\max$  columns give the minimums and maximums of  $x_{it}$  for overall,  $\bar{x}_i$  for between, and  $x_{it} - \bar{x}_i + \bar{x}$  for within.

In the `xtsum` output, Stata uses lowercase  $n$  to denote the number of individuals and uppercase  $N$  to denote the total number of individual-time observations. In our notation, these quantities are, respectively,  $N$  and  $\sum_{i=1}^N T_i$ .

The `xttab` command tabulates data in a way that provides additional details on the within and between variation of a variable. For example,

```
. * Panel tabulation for a variable
. xttab south
```

south	Overall		Between		Within Percent
	Freq.	Percent	Freq.	Percent	
0	2956	70.97	428	71.93	98.66
1	1209	29.03	182	30.59	94.90
Total	4165	100.00	610 (n = 595)	102.52	97.54

The overall summary shows that 72% of the 4,165 individual-year observations had `south` = 0 and 31% had `south` = 1. The between summary indicates that of the 595 people, 72% had `south` = 0 at least once and 31% had `south` = 1 at least once. The between total percentage is 102.52 because 2.52% of the sampled individuals (15 persons) lived some of the time in the south and some not in the south and hence are double counted. The within summary indicates that 95% of people who ever lived in the south always lived in the south during the time period covered by the panel and 99% who lived outside the south always lived outside the south. The `south` variable is close to time invariant.

The `xttab` command is most useful when the variable takes on few values because then there are few values to tabulate and interpret.

The `xttrans` command provides transition probabilities from one period to the next. For example,

```
. * Transition probabilities for a variable
. xttrans south, freq
```

Residence; south==1 if in the South area	Residence; south==1 if in the South area		Total
	0	1	
0	2,527 99.68	8 0.32	2,535 100.00
1	8 0.77	1,027 99.23	1,035 100.00
Total	2,535 71.01	1,035 28.99	3,570 100.00

One time period is lost in calculating transitions, so 3,570 observations are used. For time-invariant data, the diagonal entries will be 100%, and the off-diagonal entries will be 0%. For `south`, 99.2% of the observations ever in the south for one period remain in the south for the next period. And for those who did not live in the south for one period, 99.7% remained outside the south for the next period. The `south` variable is close to time invariant.

The `xttrans` command is most useful when the variable takes on few values.

### 8.3.5 Time-series plots for each individual

It can be useful to provide separate time-series plots for some or all individual units.

Separate time-series plots of a variable for one or more individuals can be obtained by using the `xtline` command. The `overlay` option overlays the plots for each individual on the same graph. For example,

```
. quietly xtline lwage if id<=20, overlay
```

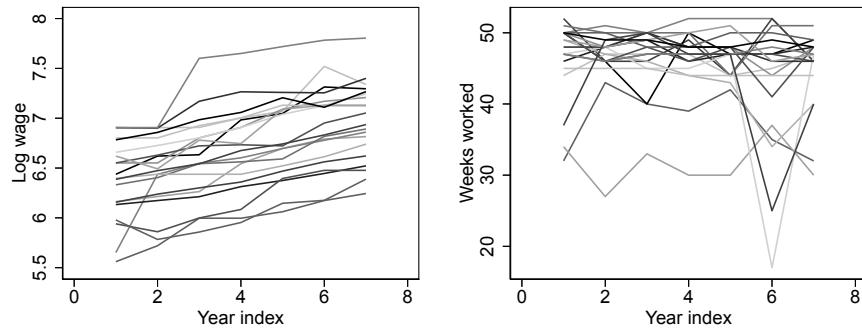
produces overlaid time-series plots of `lwage` for the first 20 individuals in the sample.

We provide time-series plots for the first 20 individuals in the sample. The default is to provide a graph legend that identifies each individual that

appears in the graph and takes up much of the graph if the graph uses data from many individuals. This legend can be suppressed by using the `legend(off)` option. Separate plots are obtained for `lwage` and for `wks`, and these are then combined by using the `graph combine` command. We have

```
. * Simple time-series plot for each of 20 individuals
. qui xtline lwage if id<=20, overlay legend(off) saving(graph1.gph, replace)
. qui xtline wks if id<=20, overlay legend(off) saving(graph2.gph, replace)
. graph combine graph1.gph graph2.gph, iscale(1.4) ysize(2.5) xsize(6.0)
```

Figure 8.1 shows that the wage rate increases roughly linearly over time, aside from two individuals with large increases from years 1 to 2, and that weeks worked show no discernible trend over time.



**Figure 8.1.** Time-series plots of log wage against year and weeks worked against year for each of the first 20 observations

### 8.3.6 Overall scatterplot

In cases where there is one key regressor, we can begin with a scatterplot of the dependent variable on the key regressor, using data from all panel observations.

The following command adds fitted quadratic regression and lowess regression curves to the scatterplot.

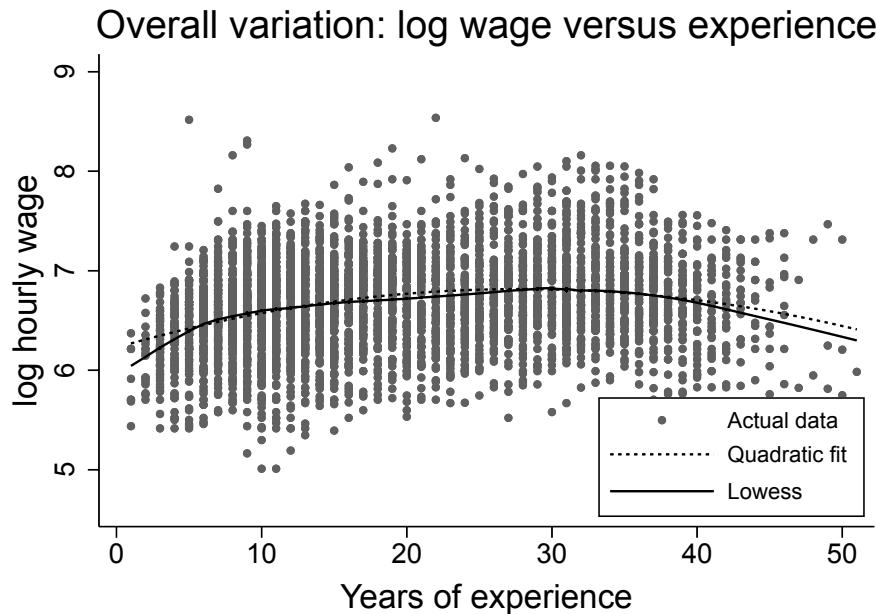
```
. graph twoway (scatter lwage exp) (qfit lwage exp) (lowess lwage exp)
```

This produces a graph that is difficult to read because the scatterplot points are very large, making it hard to then see the regression curves.

The following code presents a better-looking scatterplot of `lnwage` on `exp`, along with the fitted regression lines. It uses the same graph options as those explained in section [2.6.6](#). We have

```
. * Scatterplot, quadratic fit, and nonparametric regression (lowess)
. graph twoway (scatter lwage exp, msize(small) msymbol(o))
>   (qfit lwage exp, clstyle(p3) lwidth(medthick))
>   (lowess lwage exp, bwidth(0.4) clstyle(p1) lwidth(medthick)),
>   plotregion(style(none)) scale(1.2)
>   title("Overall variation: log wage versus experience")
>   xtitle("Years of experience", size(medlarge)) xscale(titlegap(*5))
>   ytitle("log hourly wage", size(medlarge)) yscale(titlegap(*5))
>   legend(pos(4) ring(0) col(1)) legend(size(small))
>   legend(label(1 "Actual data") label(2 "Quadratic fit") label(3 "Lowess"))
```

Each point on figure [8.2](#) represents an individual–year pair. The dashed smooth curve line is fit by OLS of `lwage` on a quadratic in `exp` (using `qfit`), and the solid line is fit by nonparametric regression (using `lowess`). Log wage increases until 30 or so years of experience and then declines.



**Figure 8.2.** Overall scatterplot of log wage against experience using all observations

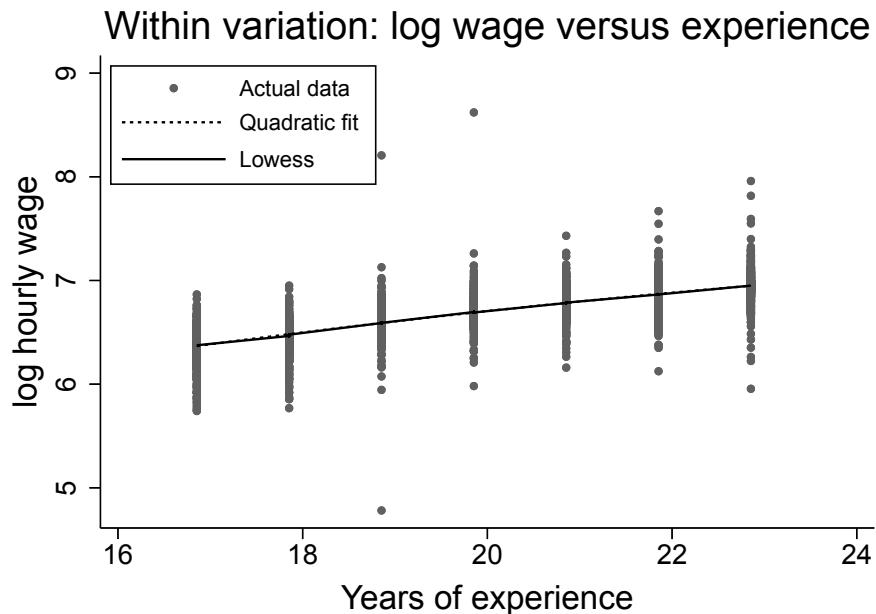
### 8.3.7 Within scatterplot

The `xtdata` command can be used to obtain similar plots for within variation, using option `fe`; between variation, using option `be`; and RE variation (the default), using option `re`. The `xtdata` command replaces the data in memory with the specified transform, so you should first `preserve` the data and then `restore` the data when you are finished with the transformed data.

For example, the `fe` option creates deviations from means, so that  $(y_{it} - \bar{y}_i + \bar{y})$  is plotted against  $(x_{it} - \bar{x}_i + \bar{x})$ . For `lwage` plotted against `exp`, we obtain

```
. * Scatterplot for within variation
. preserve
. xtdata, fe
. graph twoway (scatter lwage exp, msize(small) msymbol(o))
>     (qfit lwage exp, clstyle(p3) lwidth(medthick))
>     (lowess lwage exp, bwidth(0.4) clstyle(p1) lwidth(medthick)),
>     plotregion(style(none)) scale(1.2)
>     title("Within variation: log wage versus experience")
>     xtitle("Years of experience", size(medlarge)) xscale(titlegap(*5))
>     ytitle("log hourly wage", size(medlarge)) yscale(titlegap(*5))
>     legend(pos(11) ring(0) col(1)) legend(size(small))
>     legend(label(1 "Actual data") label(2 "Quadratic fit") label(3 "Lowess"))
. restore
```

The result is given in figure 8.3. At first glance, this figure is puzzling because only seven distinct values of `exp` appear. But the panel is balanced, and `exp` (years of work experience) is increasing by exactly one each period for each individual in this sample of people who worked every year. So  $(x_{it} - \bar{x}_i)$  increases by one each period, as does  $(x_{it} - \bar{x}_i + \bar{x})$ . The latter quantity is centered on  $\bar{x} = 19.85$  (see section 8.3.1), which is the value in the middle year with  $t = 4$ . Clearly, it can be very useful to plot a figure such as this.



**Figure 8.3.** Within scatterplot of log-wage deviations from individual means against experience deviations from individual means

### 8.3.8 Pooled OLS regression with cluster-robust standard errors

A natural starting point is a pooled OLS regression for log wage using data for all individuals in all years.

We include as regressors education, weeks worked, and a quadratic in experience. Education is a time-invariant regressor, taking the same value each year for a given individual. Weeks worked is an example of a time-varying regressor. Experience is also time-varying, though it is so deterministically because the sample comprises people who work full time in all years, so experience increases by one year as  $t$  increments by one.

Regressing  $y_{it}$  on  $\mathbf{x}_{it}$  yields consistent estimates of  $\beta$  if the composite error  $u_{it}$  in the pooled model of (8.2) is uncorrelated with  $\mathbf{x}_{it}$ . As explained in section 8.2, the error  $u_{it}$  is likely to be correlated over time for a given individual, so we use cluster-robust standard errors that cluster on the individual. We have

```

. * Pooled OLS with cluster--robust standard errors
. use mus208psid, clear
(A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e)
. regress lwage exp exp2 wks ed, vce(cluster id)

Linear regression                                         Number of obs      =     4,165
                                                               F(4, 594)        =     72.58
                                                               Prob > F       =     0.0000
                                                               R-squared        =     0.2836
                                                               Root MSE         =     .39082
(Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
exp	.044675	.0054385	8.21	0.000	.0339941 .055356
exp2	-.0007156	.0001285	-5.57	0.000	-.0009679 -.0004633
wks	.005827	.0019284	3.02	0.003	.0020396 .0096144
ed	.0760407	.0052122	14.59	0.000	.0658042 .0862772
_cons	4.907961	.1399887	35.06	0.000	4.633028 5.182894

The output shows that  $R^2 = 0.28$ , and the estimates imply that wages increase with experience until a peak at 31 years [ $= 0.0447/(2 \times 0.00072)$ ] and then decline. Wages increase by 0.6% with each additional week worked. And wages increase by 7.6% with each additional year of education.

For panel data, it is essential that OLS standard errors be corrected for clustering on the individual. In contrast, the default standard errors assume that the regression errors are independent and identically distributed (i.i.d.). Using the default standard errors, we obtain

Source	SS	df	MS	Number of obs	=	4,165
Model	251.491445	4	62.8728613	F(4, 4160)	=	411.62
Residual	635.413457	4,160	.152743619	Prob > F	=	0.0000
Total	886.904902	4,164	.212993492	R-squared	=	0.2836
				Adj R-squared	=	0.2829
				Root MSE	=	.39082

lwage	Coefficient	Std. err.	t	P> t	[95% conf. interval]
exp	.044675	.0023929	18.67	0.000	.0399838 .0493663
exp2	-.0007156	.0000528	-13.56	0.000	-.0008191 -.0006121
wks	.005827	.0011827	4.93	0.000	.0035084 .0081456
ed	.0760407	.0022266	34.15	0.000	.0716754 .080406
_cons	4.907961	.0673297	72.89	0.000	4.775959 5.039963

These standard errors are misleadingly small; the cluster–robust standard errors are, respectively, 0.0054, 0.0001, 0.0019, and 0.0052.

It is likely that if `log wage` is overpredicted in one year for a given person, then it is likely to be overpredicted in other years. Failure to control for this error correlation leads to underestimation of standard errors because, intuitively, each additional observation for a given person actually provides less than an independent piece of new information.

The difference between default and cluster–robust standard errors for pooled OLS can be very large. The difference increases with increasing  $T$ , increasing autocorrelation in model errors, and increasing autocorrelation of the regressor of interest. Specifically, from section 3.4.6, the standard error inflation factor  $\tau \simeq \sqrt{1 + \rho_u \rho_x (T - 1)}$ , where  $\rho_u$  is the intraclass correlation of the error, defined below in (8.4), and  $\rho_x$  is the intraclass correlation of the regressor. Here  $\rho_u \simeq 0.80$ , shown below, and for time-invariant regressor `ed`,  $\rho_x = 1$ , so  $\tau \simeq \sqrt{1 + 0.80 \times 1 \times 6} = \sqrt{5.8} \simeq 2.41$  for `ed`. Similarly, the regressor `exp` has  $\rho_x$  close to 1 because for this sample, experience increases by one year as  $t$  increments by 1.

Cluster–robust standard errors require that  $N \rightarrow \infty$  and that errors are independent over  $i$ . The assumption of independence over  $i$  can be relaxed to independence at a more aggregated level, provided that the number of units is still large and the units nest the individual. For example, the Panel Study of Income Dynamics is a household survey, and errors for individuals from the same household may be correlated. If, say, `houseid` is available as a household identifier, then we would use the `vce(cluster houseid)` option. As a second example, if the regressor of interest is aggregated at the state level, such as a state policy variable, and there are many states, then it may be better to use the `vce(cluster state)` option.

### 8.3.9 Time-series autocorrelations for panel data

The Stata time-series operators can be applied to panel data when both panel and time identifiers are set with the `xtset` command. Examples include `L.lwage` or `L1.lwage` for `lwage` lagged once, `L2.lwage` for `lwage` lagged twice, `D.lwage` for the difference in `lwage` (equals `lwage - L.lwage`),

`L.D.lwage` for this difference lagged once, and `L2D.lwage` for this difference lagged twice.

Use of these operators is the best way to create lagged variables because relevant missing values are automatically and correctly created. For example, `regress lwage L2.wage` will use  $(7 - 2) \times 595$  observations because forming `L2.wage` leads to a loss of the first 2 years of data for each of the 595 individuals.

The `corrgram` command for computing autocorrelations of time-series data does not work for panel data. Instead, autocorrelations can be obtained by using the `correlate` command. For example,

```
. * First-order autocorrelation in a variable
. sort id t
. correlate lwage L.lwage
(obs=3,570)
```

	L.	
	lwage	lwage
lwage		
--.	1.0000	
L1.	0.9189	1.0000

calculates the first-order autocorrelation coefficient for `lwage` to be 0.92.

We now calculate autocorrelations at all lags (here up to 6 periods). Rather than doing so for `lwage`, we do so for the residuals from the previous pooled OLS regression for `lwage`. We have

```
. * Autocorrelations of residual
. qui regress lwage exp exp2 wks ed, vce(cluster id)
. predict uhat, residuals
. forvalues j = 1/6 {
2.     qui corr uhat L`j'.uhat
3.     display "Autocorrelation at lag `j' = " %6.3f r(rho)
4. }
Autocorrelation at lag 1 =  0.884
Autocorrelation at lag 2 =  0.838
Autocorrelation at lag 3 =  0.811
Autocorrelation at lag 4 =  0.786
Autocorrelation at lag 5 =  0.750
Autocorrelation at lag 6 =  0.729
```

The `forvalues` loop leads to separate computation of each autocorrelation to maximize the number of observations used. If instead we gave a one-line command to compute the autocorrelations of `uhat` through `L6.uhat`, then only 595 observations would have been used. Here  $6 \times 595$  observations are used to compute the autocorrelation at lag 1,  $5 \times 595$  observations are used to compute the autocorrelation at lag 2, and so on. The average of the autocorrelations, 0.80, provides a rough estimate of the intraclass correlation coefficient of the residuals.

Clearly, the errors are serially correlated, and cluster-robust standard errors after pooled OLS are required. The individual-effects model provides an explanation for this correlation. If the error  $u_{it} = \alpha_i + \varepsilon_{it}$ , then even if  $\varepsilon_{it}$  is i.i.d.  $(0, \sigma_\varepsilon^2)$ , we have  $\text{Cor}(u_{it}, u_{is}) \neq 0$  for  $t \neq s$  if  $\alpha_i \neq 0$ . The individual effect  $\alpha_i$  induces correlation over time for a given individual.

The preceding estimated autocorrelations are constant across years. For example, the correlation of `uhat` with `L.uhat` across years 1 and 2 is assumed to be the same as that across years 2 and 3, years 3 and 4, ..., years 6 and 7. This presumes that the errors are stationary.

In the nonstationary case, the autocorrelations will differ across pairs of years. For example, we consider the autocorrelations one year apart and allow these to differ across the year pairs. We have

```
. * First-order autocorrelation differs in different year pairs
. forvalues s = 2/7 {
  2.      qui corr uhat L1.uhat if t == `s'
  3.      display "Autocorrelation at lag 1 in year `s' = " %6.3f r(rho)
  4.  }
Autocorrelation at lag 1 in year 2 =  0.915
Autocorrelation at lag 1 in year 3 =  0.799
Autocorrelation at lag 1 in year 4 =  0.855
Autocorrelation at lag 1 in year 5 =  0.867
Autocorrelation at lag 1 in year 6 =  0.894
Autocorrelation at lag 1 in year 7 =  0.893
```

The lag-1 autocorrelations for individual-year pairs range from 0.80 to 0.92, and their average is 0.87. From the earlier output, the lag-1 autocorrelation equals 0.88 when it is constrained to be equal across all year pairs. It is common to impose equality for simplicity.

### 8.3.10 Error correlation in the RE model

For the individual-effects model (8.1), the combined error  $u_{it} = \alpha_i + \varepsilon_{it}$ . The RE model assumes that  $\alpha_i$  is i.i.d. with a variance of  $\sigma_\alpha^2$  and that  $u_{it}$  is i.i.d. with a variance of  $\sigma_\varepsilon^2$ .

Then  $u_{it}$  has a variance of  $\text{Var}(u_{it}) = \sigma_\alpha^2 + \sigma_\varepsilon^2$  and a covariance of  $\text{Cov}(u_{it}, u_{is}) = \sigma_\alpha^2$ ,  $s \neq t$ . It follows that in the RE model,

$$\rho_u = \text{Cor}(u_{it}, u_{is}) = \sigma_\alpha^2 / (\sigma_\alpha^2 + \sigma_\varepsilon^2), \quad \text{for all } s \neq t \quad (8.4)$$

This constant correlation is called the intraclass correlation of the error.

The RE model therefore permits serial correlation in the model error. This correlation can approach 1 if the random effect is large relative to the idiosyncratic error, so that  $\sigma_\alpha^2$  is large relative to  $\sigma_\varepsilon^2$ .

This serial correlation is restricted to be the same at all lags, and the errors  $u_{it}$  are then called equicorrelated or exchangeable. From section 8.3.9, the error correlations were, respectively, 0.88, 0.84, 0.81, 0.79, 0.75, and 0.73, so a better model may be one that allows the error correlation to decrease with the lag length.

## 8.4 Pooled or population-averaged estimators

Pooled estimators simply regress  $y_{it}$  on an intercept and  $\mathbf{x}_{it}$ , using both between (cross-section) and within (time-series) variation in the data.

Standard errors need to adjust for any error correlation and, given a model for error correlation, more efficient FGLS estimation is possible. Pooled estimators, called PA estimators in the statistics literature, are consistent if the RE model is appropriate and are inconsistent if the FE model is appropriate.

### 8.4.1 Pooled OLS estimator

The pooled OLS estimator can be motivated from the individual-effects model by rewriting (8.1) as the pooled model

$$y_{it} = \alpha + \mathbf{x}'_{it}\boldsymbol{\beta} + (\alpha_i - \alpha + \varepsilon_{it}) \quad (8.5)$$

Any time-specific effects are assumed to be fixed and already included as time dummies in the regressors  $\mathbf{x}_{it}$ . The model (8.5) explicitly includes a common intercept, and the individual effects  $\alpha_i - \alpha$  are now centered on zero.

Consistency of OLS requires that the error term  $(\alpha_i - \alpha + \varepsilon_{it})$  be uncorrelated with  $\mathbf{x}_{it}$ . So pooled OLS is consistent in the RE model but is inconsistent in the FE model because then  $\alpha_i$  is correlated with  $\mathbf{x}_{it}$ .

The pooled OLS estimator for our data example has already been presented in section 8.3.8. As emphasized there, cluster-robust standard errors are necessary in the common case of a short panel with independence across individuals.

### 8.4.2 Pooled FGLS estimator or PA estimator

Pooled feasible generalized least-squares (PFGLS) estimation can lead to estimators of the parameters of the pooled model (8.5) that are more efficient

than OLS estimation. Again, we assume that any individual-level effects are uncorrelated with regressors, so PFGLS is consistent.

Different assumptions about the correlation structure for the errors  $u_{it}$  lead to different PFGLS estimators. In section [9.5](#), we present some estimators for long panels, using the `xtgls` and `xtregar` commands.

Here we consider only short panels with errors independent across individuals. We need to model the  $T \times T$  matrix of error correlations. An assumed correlation structure, called a working matrix, is specified, and the appropriate PFGLS estimator is obtained. To guard against the working matrix being a misspecified model of the error correlation, we compute cluster-robust standard errors. Better models for the error correlation lead to more efficient estimators, but the use of robust standard errors means that the estimators are not presumed to be fully efficient.

In the statistics literature, the pooled approach is called a PA approach, because any individual effects are assumed to be random and are averaged out. The PFGLS estimator is then called the PA estimator.

### 8.4.3 The `xtreg, pa` command

The pooled estimator, or PA estimator, is obtained by using the `xtreg` command (see section [8.2.4](#)) with the `pa` option. The two key additional options are `corr()`, to place different restrictions on the error correlations, and `vce(robust)`, to obtain cluster-robust standard errors that are valid even if `corr()` does not specify the correct correlation model, provided that observations are independent over  $i$  and  $N \rightarrow \infty$ .

Let  $\rho_{ts} = \text{Cor}(u_{it}, u_{is})$ , the error correlation over time for individual  $i$ , and note the restriction that  $\rho_{ts}$  does not vary with  $i$ . The `corr()` options all set  $\rho_{tt} = 1$  but differ in the model for  $\rho_{ts}$  for  $t \neq s$ . With  $T$  time periods, the correlation matrix is  $T \times T$ , and there are potentially as many as  $T(T - 1)/2$  unique off-diagonal entries.

The `corr(independent)` option sets  $\rho_{ts} = 0$  for  $s \neq t$ . Then the PA estimator equals the pooled OLS estimator.

The `corr(exchangeable)` option sets  $\rho_{ts} = \rho$  for all  $s \neq t$  so that errors are assumed to be equicorrelated. This assumption is imposed by the RE model (see section 8.3.10), and as a result, `xtreg, pa` with this option is asymptotically equivalent to `xtreg, re`.

For panel data, it is often the case that the error correlation  $\rho_{ts}$  declines as the time difference  $|t - s|$  increases—the application in section 8.3.9 provided an example. The `corr(ar k)` option models this dampening by assuming an autoregressive process of order  $k$ , or AR( $k$ ) process, for  $u_{it}$ . For example, `corr(ar 1)` assumes that  $u_{it} = \rho_1 u_{i,t-1} + \varepsilon_{it}$ , which implies that  $\rho_{ts} = \rho_1^{|t-s|}$ . The `corr(stationary g)` option instead uses a moving-average process, or MA( $g$ ) process. This sets  $\rho_{ts} = \rho_{|t-s|}$  if  $|t - s| \leq g$  and  $\rho_{ts} = 0$  if  $|t - s| > g$ .

The `corr(unstructured)` option places no restrictions on  $\rho_{ts}$ , aside from equality of  $\rho_{i,ts}$  across individuals. Then

$\text{Cov}(u_{it}, u_{is}) = 1/N \sum_i (\widehat{u}_{it} - \overline{\widehat{u}_t})(\widehat{u}_{is} - \overline{\widehat{u}_s})$ . For small  $T$ , this may be the best model, but for larger  $T$ , the method can fail numerically because there are  $T(T - 1)/2$  unique parameters  $\rho_{ts}$  to estimate. The `corr(nonstationary g)` option allows  $\rho_{ts}$  to be unrestricted if  $|t - s| \leq g$  and sets  $\rho_{ts} = 0$  if  $|t - s| > g$ , so there are fewer correlation parameters to estimate.

The PA estimator is also called the generalized estimating equations estimator in the statistics literature. The `xtreg, pa` command is the special case of `xtgee` with the `family(gaussian)` option. The more general `xtgee` command, presented in section 22.4.4, has other options that permit application to a wide range of nonlinear panel models.

#### 8.4.4 Application of the `xtreg, pa` command

As an example, we specify an AR(2) error process. We have

```

. * Population-averaged or pooled FGLS estimator with AR(2) error
. xtreg lwage exp exp2 wks ed, pa corr(ar 2) vce(robust) nolog

GEE population-averaged model
Group and time vars: id t
Family: Gaussian
Link: Identity
Correlation: AR(2)

Number of obs      =  4,165
Number of groups =    595
Obs per group:
min =          7
avg =         7.0
max =          7
Wald chi2(4)     = 873.28
Prob > chi2      = 0.0000
Scale parameter = .1966639
(Std. err. adjusted for clustering on id)

```

lwage	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
exp	.0718915	.003999	17.98	0.000	.0640535 .0797294
exp2	-.0008966	.0000933	-9.61	0.000	-.0010794 -.0007137
wks	.0002964	.0010553	0.28	0.779	-.001772 .0023647
ed	.0905069	.0060161	15.04	0.000	.0787156 .1022982
_cons	4.526381	.1056897	42.83	0.000	4.319233 4.733529

The coefficients change considerably compared with the coefficients from pooled OLS. The cluster-robust standard errors are smaller than those from pooled OLS for all regressors except `ed`, illustrating the desired improved efficiency because of better modeling of the error correlations. Note that unlike the pure time-series case, controlling for autocorrelation does not lead to the loss of initial observations.

The estimated correlation matrix is stored in `e(R)`. We have

```

. * Estimated error correlation matrix after xtreg, pa
. matrix list e(R)

symmetric e(R) [7,7]
      c1        c2        c3        c4        c5        c6        c7
r1       1
r2   .89722058      1
r3   .84308581   .89722058      1
r4   .78392846   .84308581   .89722058      1
r5   .73064474   .78392846   .84308581   .89722058      1
r6   .6806209   .73064474   .78392846   .84308581   .89722058      1
r7   .63409777   .6806209   .73064474   .78392846   .84308581   .89722058      1

```

By comparison, from section [8.3.9](#), the autocorrelations of the errors after pooled OLS estimation were 0.88, 0.84, 0.81, 0.79, 0.75, and 0.73.

In an end-of-chapter exercise, we compare estimates obtained using different error-correlation structures.

## 8.5 Fixed-effects or within estimator

Estimators of the parameters  $\beta$  of the FE model (8.1) must remove the fixed effects  $\alpha_i$ . The within transform does so by mean-differencing. The within estimator performs OLS on the mean-differenced data. Because all the observations of the mean-difference of a time-invariant variable are zero, we cannot estimate the coefficient on a time-invariant variable.

Because the within estimator provides a consistent estimate of the FE model, it is often called the FE estimator, though the first-difference (FD) estimator given in section 8.9 also provides consistent estimates in the FE model. The within estimator is also consistent under the RE model, but alternative estimators are more efficient in the RE model.

### 8.5.1 Within estimator

The fixed effects  $\alpha_i$  in the model (8.1) can be eliminated by subtraction of the corresponding model for individual means  $\bar{y}_i = \bar{\mathbf{x}}_i' \beta + \bar{\varepsilon}_i$ , leading to the within model or mean-difference model

$$(y_{it} - \bar{y}_i) = (\mathbf{x}_{it} - \bar{\mathbf{x}}_i)' \beta + (\varepsilon_{it} - \bar{\varepsilon}_i) \quad (8.6)$$

where, for example,  $\bar{\mathbf{x}}_i = T_i^{-1} \sum_{t=1}^{T_i} \mathbf{x}_{it}$ , and only regressors that have within variation are included in (8.6) because otherwise  $x_{it} - \bar{x}_{it} = 0$ . The within estimator is the OLS estimator of this model.

Because  $\alpha_i$  has been eliminated, OLS leads to consistent estimates of  $\beta$  even if  $\alpha_i$  is correlated with  $\mathbf{x}_{it}$ , as is the case in the FE model. This result is a great advantage of panel data. Consistent estimation is possible even with endogenous regressors  $\mathbf{x}_{it}$ , provided that  $\mathbf{x}_{it}$  is correlated only with the time-invariant component of the error,  $\alpha_i$ , and not with the time-varying component of the error,  $\varepsilon_{it}$ .

This desirable property of consistent parameter estimation in the FE model is tempered, however, by the inability to estimate the coefficients or a

time-invariant regressor. Also the within estimator will be relatively imprecise for time-varying regressors that vary little over time.

Stata actually fits the model

$$(y_{it} - \bar{y}_i + \bar{\bar{y}}) = \alpha + (\mathbf{x}_{it} - \bar{\mathbf{x}}_i + \bar{\bar{\mathbf{x}}})' \boldsymbol{\beta} + (\varepsilon_{it} - \bar{\varepsilon}_i + \bar{\bar{\varepsilon}}) \quad (8.7)$$

where, for example,  $\bar{\bar{y}} = (1/N)\bar{y}_i$  is the grand mean of  $y_{it}$ . This parameterization has the advantage of providing an intercept estimate, the average of the individual effects  $\alpha_i$ , while yielding the same slope estimate  $\boldsymbol{\beta}$  as that from the within model.

### 8.5.2 The xtreg, fe command

The within estimator is computed by using the `xtreg` command (see section 8.2.4) with the `fe` option. The default standard errors assume that after one controls for  $\alpha_i$ , the error  $\varepsilon_{it}$  is i.i.d. The `vce(robust)` option, equivalent to `vce(cluster id)`, relaxes this assumption and provides cluster-robust standard errors, provided that observations are independent over  $i$  and  $N \rightarrow \infty$ .

### 8.5.3 Application of the xtreg, fe command

For our data, we obtain

```

. * Within or FE estimator with cluster--robust standard errors
. xtreg lwage exp exp2 wks ed, fe vce(cluster id)
note: ed omitted because of collinearity.

Fixed-effects (within) regression                               Number of obs     =      4,165
Group variable: id                                         Number of groups  =       595
R-squared:                                                 Obs per group:
    Within  = 0.6566                                         min  =         7
    Between = 0.0276                                         avg  =      7.0
    Overall = 0.0476                                         max  =         7
                                                F(3,594)          =   1059.72
corr(u_i, Xb) = -0.9107                                     Prob > F        =     0.0000
                                                               (Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
exp	.1137879	.0040289	28.24	0.000	.1058753 .1217004
exp2	-.0004244	.0000822	-5.16	0.000	-.0005858 -.0002629
wks	.0008359	.0008697	0.96	0.337	-.0008721 .0025439
ed	0	(omitted)			
_cons	4.596396	.0600887	76.49	0.000	4.478384 4.714408
sigma_u	1.0362039				
sigma_e	.15220316				
rho	.97888036	(fraction of variance due to u_i)			

Compared with the standard errors of pooled OLS, the standard errors here have roughly tripled because only within variation of the data is being used. The `sigma_u` and `sigma_e` entries are explained in section [8.8.1](#), and the  $R^2$  measures are explained in section [8.8.2](#).

It is imperative to note that inclusion of fixed effects controls for only some of the within cluster correlation. From output not included, the corresponding default standard errors are roughly 40% too small. They equal, respectively, 0.00247, 0.00005, 0.00061, and 0.03891.

A striking result is that the coefficient for education is not identified. This is because the data on education is time invariant. In fact, given that we knew from the `xtsum` output in section [8.3.4](#) that `ed` had zero within standard deviation, we should not have included it as one of the regressors in the `xtreg, fe` command.

This is unfortunate because how wages depend on education is of great policy interest. It is certainly endogenous because people with high ability are likely to have on average both high education and high wages. Alternative panel-data methods to control for endogeneity of the `ed` variable are presented in chapter 9. In other panel applications, endogenous regressors may be time varying, and the within estimator will suffice.

### 8.5.4 Least-squares dummy-variable regression

It is best to use the `xtreg, fe` command to fit the FE model. For completeness, we present an alternative method for obtaining consistent estimates of  $\beta$  in the FE model.

The Frisch–Waugh–Lovell theorem states that for the model  $y = \mathbf{X}_1\beta_1 + \mathbf{X}_2\beta_2 + u$ , the estimate  $\hat{\beta}_2$  obtained from OLS regression of  $y$  on both  $\mathbf{X}_1$  and  $\mathbf{X}_2$  is equivalent to the estimate  $\hat{\beta}_2$  obtained by OLS regression of  $y$  on just  $\tilde{\mathbf{X}}_2$ , where  $\tilde{\mathbf{X}}_2$  is a matrix of residuals obtained from OLS regression of each column of  $\mathbf{X}_2$  on  $\mathbf{X}_1$ .

The least-squares dummy-variable (LSDV) model introduces  $N$  individual-specific indicator variables (or dummy variables)  $d_{j,it}$ ,  $j = 1, \dots, N$ , where  $d_{j,it} = 1$  for the  $it$ th observation if  $j = 1$  and  $d_{j,it} = 0$  otherwise. Then, fit by OLS the model

$$y_{it} = \left( \sum_{j=1}^N \alpha_j d_{j,it} \right) + \mathbf{x}'_{it}\beta + \varepsilon_{it} \quad (8.8)$$

By the Frisch–Waugh–Lovell theorem, the resulting OLS estimate of  $\hat{\beta}$  can equivalently be found by first OLS regressing each component of  $\mathbf{x}_{it}$  on  $d_{1,it}, \dots, d_{N,it}$ , and then obtaining  $\hat{\beta}$  by OLS regression of  $y_{it}$  on these residuals. But these residuals can be shown to equal  $\mathbf{x}_{it} - \bar{\mathbf{x}}_i$ , so the LSDV estimate of  $\beta$  is equivalent to the within estimator. (The terminology “FE estimator” arises because in the LSDV method, the individual-specific effects  $\alpha_i$  are treated as fixed quantities to be estimated.)

Direct estimation of (8.8) is unnecessarily computationally expensive because an  $(N + K) \times (N + K)$  matrix needs to be inverted. The `areg` command with option `absorb()` avoids this by implementing the Frisch–Waugh–Lovell method, so it reports only the estimates of the parameters  $\beta$ . We have

```
. * LSDV model fit using areg with cluster--robust standard errors
. areg lwage exp exp2 wks ed, absorb(id) vce(cluster id)
note: ed omitted because of collinearity.

Linear regression, absorbing indicators
Absorbed variable: id

Number of obs      =  4,165
No. of categories =   595
F(3, 594)          = 908.44
Prob > F           = 0.0000
R-squared           = 0.9068
Adj R-squared       = 0.8912
Root MSE            = 0.1522

(Std. err. adjusted for 595 clusters in id)
```

lwage	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
exp	.1137879	.0043514	26.15	0.000	.1052418	.1223339
exp2	-.0004244	.0000888	-4.78	0.000	-.0005988	-.00025
wks	.0008359	.0009393	0.89	0.374	-.0010089	.0026806
ed	0	(omitted)				
_cons	4.596396	.0648993	70.82	0.000	4.468936	4.723856

The coefficient estimates are the same as those from `xtreg, fe`. Because of different small-sample correction (see section 6.6.4), the cluster-robust standard errors following `areg` are approximately  $\sqrt{T/(T - 1)} = \sqrt{7/6}$  times larger than those from `xtreg, fe`, and the `xtreg, fe` standard errors should be used. This difference arises because inference for `areg` is designed for the case where  $N$  is fixed and  $T \rightarrow \infty$ , whereas we are considering the short-panel case, where  $T$  is fixed and  $N \rightarrow \infty$ .

For completeness, we use the `regress` command to directly fit the LSDV model (8.8) by including a set of indicator variables for each individual by inserting the `i.` operator before the categorical variable `id`. Because there are  $N + K$  regressors, the default setting of `matsize` may need to be increased if Stata 15 or earlier is used, and the output from `regress` is very long because it includes coefficients for all the dummy variables. We instead

suppress the output and use `estimates table` to list results for just the coefficients of interest.

```
. * LSDV model fit using factor variables with cluster--robust standard errors
. qui regress lwage exp exp2 wks ed i.id, vce(cluster id)
. estimates table, keep(exp exp2 wks ed _cons) b se b(%12.7f)
```

Variable	Active
exp	0.1137879
	0.0043514
exp2	-0.0004244
	0.0000888
wks	0.0008359
	0.0009393
ed	0.1022134
	0.0046744
_cons	4.3476807
	0.0443191

Legend: b/se

The coefficient estimates and standard errors are exactly the same as those obtained from `areg`, aside from the constant. For `areg` (and `xtreg, fe`), the intercept is fit so that  $\bar{y} - \bar{x}'\hat{\beta} = 0$ , whereas this is not the case using `regress`. The cluster-robust standard errors are the same as those from `areg`, and as already noted, those from `xtreg, fe` should be used.

The complete output includes estimates  $\hat{\alpha}_i = \bar{y}_i - \bar{x}'_i\hat{\beta}$ ,  $i = 1, \dots, N$ . Note that  $\hat{\alpha}_i$  is not consistently estimated in short panels, because it essentially relies on only  $T_i$  observations used to form  $\bar{y}_i$  and  $\bar{x}_i$ , but  $\hat{\beta}$  is nonetheless consistently estimated.

For a short panel, it is much slower to use the LSDV method because it requires inverting an  $(N + K) \times (N + K)$  matrix, whereas `xtreg, fe` or `areg` inverts a much smaller  $K \times K$  matrix. And, as noted, `xtreg, fe` gives correct cluster-robust standard errors.

### 8.5.5 Two-way fixed effects

For short panels, the most common case of two-way fixed effects is to have individual fixed effects and time fixed effects. Then simply give a command

such as `xtreg y x i.time, fe`, where `time` is the time variable. Then a  $(K + T) \times (K + T)$  matrix needs to be inverted because the FE estimator eliminates the  $N$  individual fixed effects.

In some cases, there may be need to add many fixed effects, in addition to the individual fixed effects. Then the methods and community-contributed commands such as `reg2hdfe` and `felsdvreg` presented in section [6.6.6](#) may be used.

## 8.6 Between estimator

The between estimator uses only between or cross-sectional variation in the data and is the OLS estimator from the regression of  $\bar{y}_i$  on  $\bar{\mathbf{x}}_i$ . Because only cross-sectional variation in the data is used, the coefficients of any individual-invariant regressors, such as time dummies, cannot be identified. We provide the estimator for completeness, even though it is seldom used because pooled estimators and the RE estimator are more efficient.

### 8.6.1 Between estimator

The between estimator is inconsistent in the FE model and is consistent in the RE model. To see this, average the individual-effects model (8.1) to obtain the between model

$$\bar{y}_i = \alpha + \bar{\mathbf{x}}_i' \boldsymbol{\beta} + (\alpha_i - \alpha + \bar{\varepsilon}_i)$$

The between estimator is the OLS estimator in this model. Consistency requires that the error term  $(\alpha_i - \alpha + \bar{\varepsilon}_i)$  be uncorrelated with  $\mathbf{x}_{it}$ . This is the case if  $\alpha_i$  is a random effect but not if  $\alpha_i$  is a fixed effect.

### 8.6.2 Application of the xtreg, be command

The between estimator is obtained by specifying the `be` option of the `xtreg` command. There is no explicit option to obtain heteroskedasticity-robust standard errors, but these can be obtained by using the `vce(bootstrap)` option.

For our data, the bootstrap standard errors differ from the default by only 10% because averages are used so that the complication is one of heteroskedastic errors rather than clustered errors. We report the default standard errors that are much more quickly computed. We have

```

. * Between estimator with default standard errors
. xtreg lwage exp exp2 wks ed, be

Between regression (regression on group means)  Number of obs      =      4,165
Group variable: id                           Number of groups     =       595
R-squared:
    Within   = 0.1357
    Between  = 0.3264
    Overall   = 0.2723
Obs per group:
    min   =        7
    avg   =      7.0
    max   =        7
F(4,590)           =     71.48
sd(u_i + avg(e_i.)) = .324656
Prob > F          = 0.0000

```

lwage	Coefficient	Std. err.	t	P> t	[95% conf. interval]
exp	.038153	.0056967	6.70	0.000	.0269647 .0493412
exp2	-.0006313	.0001257	-5.02	0.000	-.0008781 -.0003844
wks	.0130903	.0040659	3.22	0.001	.0051048 .0210757
ed	.0737838	.0048985	15.06	0.000	.0641632 .0834044
_cons	4.683039	.2100989	22.29	0.000	4.270407 5.095672

The estimates and standard errors are closer to those obtained from pooled OLS than those obtained from within estimation.

## 8.7 Random-effects estimator

The RE estimator is the FGLS estimator in the RE model (8.1) under the assumption that the random effect  $\alpha_i$  is i.i.d. and the idiosyncratic error  $\varepsilon_{it}$  is i.i.d. The RE estimator is consistent if the RE model is appropriate and is inconsistent if the FE model is appropriate.

### 8.7.1 RE estimator

The RE model is the individual-effects model (8.1)

$$y_{it} = \mathbf{x}'_{it}\beta + (\alpha_i + \varepsilon_{it}) \quad (8.9)$$

with  $\alpha_i \sim (\alpha, \sigma_\alpha^2)$  and  $\varepsilon_{it} \sim (0, \sigma_\varepsilon^2)$ . Then from (8.4), the combined error  $u_{it} = \alpha_i + \varepsilon_{it}$  is correlated over  $t$  for given  $i$  with

$$\text{Cor}(u_{it}, u_{is}) = \sigma_\alpha^2 / (\sigma_\alpha^2 + \sigma_\varepsilon^2), \quad \text{for all } s \neq t \quad (8.10)$$

The RE estimator is the FGLS estimator of  $\beta$  in (8.9) given (8.10) for the error correlations.

In several different settings, such as heteroskedastic errors and AR(1) errors, the FGLS estimator can be calculated as the OLS estimator in a model transformed to have homoskedastic uncorrelated errors. This is also possible here. Some considerable algebra shows that the RE estimator can be obtained by OLS estimation in the transformed model

$$(y_{it} - \hat{\theta}_i \bar{y}_i) = (1 - \hat{\theta}_i) \alpha + (\mathbf{x}'_{it} - \hat{\theta}_i \bar{\mathbf{x}}_i)' \beta + \left\{ (1 - \hat{\theta}_i) \alpha_i + (\varepsilon_{it} - \hat{\theta}_i \bar{\varepsilon}_i) \right\} \quad (8.11)$$

where  $\hat{\theta}_i$  is a consistent estimate of

$$\theta_i = 1 - \sqrt{\sigma_\varepsilon^2 / (T_i \sigma_\alpha^2 + \sigma_\varepsilon^2)}$$

The RE estimator is consistent and fully efficient if the RE model is appropriate. It is inconsistent if the FE model is appropriate because then correlation between  $\mathbf{x}_{it}$  and  $\alpha_i$  implies correlation between the regressors and the error in (8.11). Also, if there are no fixed effects but the errors exhibit within-panel correlation, then the RE estimator is consistent but inefficient, and cluster-robust standard errors should be obtained.

The RE estimator uses both between and within variation in the data and has special cases of pooled OLS ( $\hat{\theta}_i = 0$ ) and within estimation ( $\hat{\theta}_i = 1$ ). The RE estimator approaches the within estimator as  $T$  gets large and as  $\sigma_\alpha^2$  gets large relative to  $\sigma_\varepsilon^2$  because in those cases  $\hat{\theta}_i \rightarrow 1$ .

### 8.7.2 The xtreg, re command

Three closely related and asymptotically equivalent RE estimators can be obtained by using the `xtreg` command (see section 8.2.4) with the `re`, `mle`, or `pa` option. These estimators use different estimates of the variance components  $\sigma_\varepsilon^2$  and  $\sigma_\alpha^2$  and hence different estimates  $\hat{\theta}_i$  in the RE regression; see [XT] `xtreg` for the formulas.

The RE estimator uses unbiased estimates of the variance components and is obtained by using the `re` option. The maximum likelihood estimator, under the additional assumption of normally distributed  $\alpha_i$  and  $\varepsilon_{it}$ , is computed by using the `mle` option. The RE model implies the errors are equicorrelated or exchangeable (see section 8.3.10), so `xtreg` with the `pa` and `corr(exchangeable)` options yields asymptotically equivalent results.

For panel data, the RE estimator assumption of equicorrelated errors is usually too strong. At the least, use the `vce(cluster id)` option to obtain cluster-robust standard errors. And more efficient estimates can be obtained with `xtreg, pa` with a better error structure than those obtained with the `corr(exchangeable)` option.

### 8.7.3 Application of the xtreg, re command

For our data, `xtreg, re` yields

```

. * RE estimator with cluster--robust standard errors
. xtreg lwage exp exp2 wks ed, re vce(cluster id) theta
Random-effects GLS regression
Group variable: id
R-squared:
    Within = 0.6340
    Between = 0.1716
    Overall = 0.1830
corr(u_i, X) = 0 (assumed)
theta        = .82280511
                                         Number of obs      =     4,165
                                         Number of groups  =      595
                                         Obs per group:
                                              min =          7
                                              avg =       7.0
                                              max =          7
                                         Wald chi2(4)     =   1598.50
                                         Prob > chi2      =     0.0000
                                         (Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
exp	.0888609	.0039992	22.22	0.000	.0810227	.0966992
exp2	-.0007726	.0000896	-8.62	0.000	-.0009481	-.000597
wks	.0009658	.0009259	1.04	0.297	-.000849	.0027806
ed	.1117099	.0083954	13.31	0.000	.0952552	.1281647
_cons	3.829366	.1333931	28.71	0.000	3.567921	4.090812
sigma_u	.31951859					
sigma_e	.15220316					
rho	.81505521				(fraction of variance due to u_i)	

Unlike the within estimator, the coefficient of the time-invariant regressor `ed` is now estimated. The standard errors are somewhat smaller than those for the within estimator because some between variation is also used. The entries `sigma_u`, `sigma_e`, and `rho`, and the various  $R^2$  measures, are explained in the next section.

The `re`, `mle`, and `pa` `corr(exchangeable)` options of `xtreg` yield asymptotically equivalent estimators that differ in typical sample sizes. Comparison for these data is left as an exercise.

### 8.7.4 Correlated RE model

Some econometricians have suggested reasons for narrowing the distinction between RE and FE models by relaxing the assumption that the random effect ( $\alpha_i$ ) is purely random and uncorrelated with exogenous variables  $x_i$ . Instead, an additional assumption makes the random effect a linear function of observable exogenous variables plus an error term.

A leading example of this approach, due to [Mundlak \(1978\)](#), and often referred to as a “Mundlak correction”, assumes  $E(\alpha_i | x_{i1}, \dots, x_{iT}) = \bar{x}'_1 \gamma$ , where  $\bar{x}_{1i}$  are

the time averages of the subset of regressors that have within variation, and defines the individual specific-effect

$$\alpha_i = \bar{\mathbf{x}}_{1i}'\boldsymbol{\gamma} + \eta_i$$

where  $\eta_i$  is an independent error. Then the model (8.9) becomes

$$y_{it} = \mathbf{x}_{it}'\boldsymbol{\beta} + \bar{\mathbf{x}}_{1i}'\boldsymbol{\gamma} + (\eta_i + \varepsilon_{it})$$

This model, called a correlated RE model, is interpreted as an RE model in which the RE assumptions hold conditionally on both  $\mathbf{x}_{it}$  and  $\bar{\mathbf{x}}_{1i}$ . The addition of the extra controls could make the RE assumption more acceptable.

This approach is especially useful for those nonlinear panel models for which there is no standard FE estimator in short panels because of the incidental parameters problem; see section 22.2.1.

In the case of linear regression, RE estimation (and OLS estimation) with the Mundlak correction actually yields the same estimates as FE estimation. We have

```

. * Mundlak correction: RE with individual-specific means added as regressors
. sort id
. foreach x of varlist exp exp2 wks ed {
2.      by id: egen mean`x' = mean(`x')
3. }
. xtreg lwage exp exp2 wks ed meanexp meanexp2 meanwks meaned, re vce(cluster id)
note: meaned omitted because of collinearity.

Random-effects GLS regression
Group variable: id
Number of obs      =      4,165
Number of groups   =       595
Obs per group:
R-squared:
    Within     = 0.6566
    Between    = 0.3264
    Overall    = 0.4160
Wald chi2(7)      =  3237.73
corr(u_i, X) = 0 (assumed)
Prob > chi2        = 0.0000
(Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
exp	.1137879	.0040308	28.23	0.000	.1058876 .1216881
exp2	-.0004244	.0000823	-5.16	0.000	-.0005856 -.0002632
wks	.0008359	.0008701	0.96	0.337	-.0008695 .0025412
ed	.0737838	.0052096	14.16	0.000	.0635732 .0839943
meanexp	-.0756349	.0075336	-10.04	0.000	-.0904005 -.0608693
meanexp2	-.0002069	.0001695	-1.22	0.222	-.0005392 .0001254
meanwks	.0122544	.0043588	2.81	0.005	.0037113 .0207975
meaned	0	(omitted)			
_cons	4.683039	.2443095	19.17	0.000	4.204201 5.161877
sigma_u	.31951859				
sigma_e	.15220316				
rho	.81505521	(fraction of variance due to u_i)			

The coefficients of the individual-varying and time-varying regressors `exp`, `exp2`, and `wks` are identical to the FE estimates given in section [8.5](#), and the associated standard errors are very close. As explained earlier, the coefficient of `meaned`, the mean of the time-invariant regressor `ed`, is not identified, and this variable could have been omitted.

[Wooldridge \(2021\)](#) extends the Mundlak regression to additionally include as regressors time-period specific averages  $\bar{x}_{2t}$  and shows equivalence to the two-way FE estimator that includes both individual and time-specific fixed effects.

## 8.8 Comparison of estimators

Output from `xtreg` includes estimates of the standard deviation of the error components and  $R^2$  measures that measure within, between, and overall fit. Prediction is possible using the `predict` postestimation command. We present these estimates before turning to comparison of OLS, between, RE, and within estimators.

### 8.8.1 Estimates of variance components

Output from the `fe`, `re`, and `mle` options of `xtreg` includes estimates of the standard deviations of the error components. The combined error in the individual-effects model that we label  $\alpha_i + \varepsilon_{it}$  is referred to as  $u_i + e_{it}$  in the Stata documentation and output. Thus, Stata output `sigma_u` gives the standard deviation of the individual effect  $\alpha_i$ , and `sigma_e` gives the standard deviation of the idiosyncratic error  $\varepsilon_{it}$ .

For the RE model estimates given in the previous section, the estimated standard deviation of  $\alpha_i$  is twice that of  $\varepsilon_{it}$ . So the individual-specific component of the error (the random effect) is much more important than the idiosyncratic error.

The output labeled `rho` equals the intraclass correlation of the error  $\rho_u$  defined in (8.4). For the RE model, for example, the estimate of 0.815 is very high. This is expected because, from section 8.3.9, the average autocorrelation of the OLS residuals was computed to be around 0.80.

The `theta` option, available for the `re` option in the case of balanced data, reports the estimate  $\widehat{\theta}_i = \widehat{\theta}$ . Because  $\widehat{\theta} = 0.823$ , here the RE estimates will be much closer to the within estimates than to the OLS estimates. More generally, in the unbalanced case, the matrix `e(theta)` saves the minimum, 5th percentile, median, 95th percentile, and maximum of  $\widehat{\theta}_1, \dots, \widehat{\theta}_N$ .

### 8.8.2 Within and between $R^2$

The table header from `xtreg` provides three  $R^2$  measures, computed using the interpretation of  $R^2$  as the squared correlation between the actual and fitted values of the dependent variable, where the fitted values ignore the contribution of  $\widehat{\alpha}_i$ .

Let  $\widehat{\alpha}$  and  $\widehat{\beta}$  be estimates obtained by one of the `xtreg` options (`be`, `fe`, or `re`). Let  $r^2(x, y)$  denote the squared correlation between  $x$  and  $y$ . Then,

$$\begin{aligned} \text{Within } R^2: & \quad r^2 \left\{ (y_{it} - \bar{y}_i), \left( \mathbf{x}'_{it} \widehat{\beta} - \bar{\mathbf{x}}'_i \widehat{\beta} \right) \right\} \\ \text{Between } R^2: & \quad r^2 \left( \bar{y}_i, \bar{\mathbf{x}}'_i \widehat{\beta} \right) \\ \text{Overall } R^2: & \quad r^2 \left( y_{it}, \mathbf{x}'_{it} \widehat{\beta} \right) \end{aligned}$$

The three  $R^2$  measures are, respectively, 0.66, 0.03, and 0.05 for the within estimator; 0.14, 0.33, and 0.27 for the between estimator; and 0.63, 0.17, and 0.18 for the RE estimator. So the within estimator best explains the within variation ( $R_w^2 = 0.66$ ), and the between estimator best explains the between variation ( $R_b^2 = 0.33$ ). The within estimator has a low  $R_o^2 = 0.05$  and a much higher  $R^2 = 0.91$  in section [8.5.4](#) because  $R_o^2$  neglects  $\widehat{\alpha}_i$ .

### 8.8.3 Estimator comparison

We compare some of the panel estimators and associated standard errors, variance components estimates, and  $R^2$ . Pooled OLS is the same as the `xtreg` command with the `corr(independent)` and `pa` options. We have

```
. * Compare OLS, BE, FE, RE estimators, and methods to compute standard errors
. global xlist exp exp2 wks ed
. qui regress lwage $xlist, vce(cluster id)
. estimates store OLS_rob
. qui xtreg lwage $xlist, be
. estimates store BE
. qui xtreg lwage $xlist, fe
. estimates store FE
. qui xtreg lwage $xlist, fe vce(robust)
. estimates store FE_rob
. qui xtreg lwage $xlist, re
. estimates store RE
. qui xtreg lwage $xlist, re vce(robust)
. estimates store RE_rob
```

```
. estimates table OLS_rob BE FE FE_rob RE RE_rob,
>     b se stats(N r2 r2_o r2_b r2_w sigma_u sigma_e rho) b(%7.4f)
```

Variable	OLS_rob	BE	FE	FE_rob	RE
exp	0.0447	0.0382	0.1138	0.1138	0.0889
	0.0054	0.0057	0.0025	0.0040	0.0028
exp2	-0.0007	-0.0006	-0.0004	-0.0004	-0.0008
	0.0001	0.0001	0.0001	0.0001	0.0001
wks	0.0058	0.0131	0.0008	0.0008	0.0010
	0.0019	0.0041	0.0006	0.0009	0.0007
ed	0.0760	0.0738	(omitted)	(omitted)	0.1117
	0.0052	0.0049			0.0061
_cons	4.9080	4.6830	4.5964	4.5964	3.8294
	0.1400	0.2101	0.0389	0.0601	0.0936
N	4165	4165	4165	4165	4165
r2	0.2836	0.3264	0.6566	0.6566	
r2_o		0.2723	0.0476	0.0476	0.1830
r2_b		0.3264	0.0276	0.0276	0.1716
r2_w		0.1357	0.6566	0.6566	0.6340
sigma_u			1.0362	1.0362	0.3195
sigma_e			0.1522	0.1522	0.1522
rho			0.9789	0.9789	0.8151

Legend: b/se

Variable	RE_rob
exp	0.0889
	0.0040
exp2	-0.0008
	0.0001
wks	0.0010
	0.0009
ed	0.1117
	0.0084
_cons	3.8294
	0.1334
N	4165
r2	
r2_o	0.1830
r2_b	0.1716
r2_w	0.6340
sigma_u	0.3195
sigma_e	0.1522
rho	0.8151

Legend: b/se

Several features emerge. The estimated coefficients vary considerably across estimators, especially for the time-varying regressors. This reflects quite different results according to whether within variation or between variation is used. The within estimator did not provide a coefficient estimate for the time-invariant regressor  $\text{ed}$  (with the coefficient reported as 0.00). Cluster-robust standard errors for the FE and RE models exceed the default standard errors by one-third to one-half. The various  $R^2$  measures and variance-components estimates also vary considerably across models.

#### 8.8.4 FE versus RE

The essential distinction in microeconomics analysis of panel data is that between FE and RE models. If effects are fixed, then the pooled OLS and RE estimators are inconsistent, and instead the within (or FE) estimator needs to be used. The within estimator is otherwise less desirable because using only within variation leads to less efficient estimation and inability to estimate coefficients of time-invariant regressors.

To understand this distinction, consider the scalar regression of  $y_{it}$  on  $x_{it}$ . Consistency of the pooled OLS estimator requires that  $E(u_{it}|x_{it}) = 0$  in the model  $y_{it} = \alpha + \beta x_{it} + u_{it}$ . If this assumption fails so that  $x_{it}$  is endogenous, IV estimation can yield consistent estimates. It can be difficult to find an instrument  $z_{it}$  for  $x_{it}$  that satisfies  $E(u_{it}|z_{it}) = 0$ .

Panel data provide an alternative way to obtain consistent estimates. Introduce the individual-effects model  $y_{it} = \alpha_i + \beta x_{it} + \varepsilon_{it}$ . Consistency in this model requires the weaker assumption that  $E(\varepsilon_{it}|\alpha_i, x_{it}) = 0$ . Essentially, the error has two components: the time-invariant component  $\alpha_i$  correlated with regressors that we can eliminate through differencing and a time-varying component that, given  $\alpha_i$ , is uncorrelated with regressors.

The RE model adds an additional assumption to the individual-effects model:  $\alpha_i$  is distributed independently of  $x_{it}$ . This is a much stronger assumption because it implies that  $E(\varepsilon_{it}|\alpha_i, x_{it}) = E(\varepsilon_{it}|x_{it})$ , so consistency requires that  $E(\varepsilon_{it}|x_{it}) = 0$ , as assumed by the pooled OLS model.

For individual-effects models, the fundamental issue is whether the individual effect is correlated with regressors.

### 8.8.5 Hausman test for FE

Under the null hypothesis that individual effects are random, these estimators should be similar because both are consistent. Under the alternative, these estimators diverge. This juxtaposition is a natural setting for a Hausman test (see section [11.9](#)), comparing FE and RE estimators. The test compares the estimable coefficients of time-varying regressors or can be applied to a key subset of these (often one key regressor).

#### The `hausman` command

For completeness we begin with the `hausman` command, which implements the standard form of the Hausman test. This command should be used only if inference is based on default standard errors.

We have already stored the within estimates as `FE` and the `RE` estimates as `RE`, so we can immediately implement the test. The default version of the `hausman FE RE` command leads to a variance estimate

$\{\widehat{V}(\widehat{\beta}_{\text{FE}}) - \widehat{V}(\widehat{\beta}_{\text{RE}})\}$  that for these data is negative definite, so estimated standard errors of  $(\widehat{\beta}_{j,\text{FE}} - \widehat{\beta}_{j,\text{RE}})$  cannot be obtained. This problem can arise because different estimates of the error variance are used in forming  $\widehat{V}(\widehat{\beta}_{\text{FE}})$  and  $\widehat{V}(\widehat{\beta}_{\text{RE}})$ . Similar issues arise for a Hausman test comparing OLS and two-stage least-squares estimates.

This problem can be avoided by using the asymptotically equivalent `sigmamore` option, which specifies that both covariance matrices are based on the (same) estimated disturbance variance from the efficient estimator. We obtain

```
. * Hausman test assuming RE estimator is fully efficient under null hypothesis
. hausman FE RE, sigmamore
```

	Coefficients		(b-B) Difference	sqrt(diag(V_b-V_B)) Std. err.
	(b) FE	(B) RE		
exp	.1137879	.0888609	.0249269	.0012778
exp2	-.0004244	-.0007726	.0003482	.0000285
wks	.0008359	.0009658	-.0001299	.0001108

b = Consistent under H0 and Ha; obtained from xtreg.

B = Inconsistent under Ha, efficient under H0; obtained from xtreg.

Test of H0: Difference in coefficients not systematic

```
chi2(3) = (b-B)'[(V_b-V_B)^(-1)](b-B)
          = 1513.02
Prob > chi2 = 0.0000
```

The output from `hausman` provides a nice side-by-side comparison. For the coefficient of regressor `exp`, for example, a test of `RE` against `FE` yields  $t = 0.0249/0.00128 = 19.5$ , a highly statistically significant difference. And the overall statistic, here  $\chi^2(3)$ , has  $p = 0.000$ . This leads to strong rejection of the null hypothesis that `RE` provides consistent estimates.

### Cluster-robust Hausman test

A serious shortcoming of the standard Hausman test is that it requires the RE estimator to be efficient. This in turn requires that the  $\alpha_i$  and  $\varepsilon_{it}$  be i.i.d., an invalid assumption if cluster-robust standard errors for the RE estimator differ substantially from default standard errors. For our data example, and in many applications, a robust version of the Hausman test is needed. There is no official Stata command for this. A panel bootstrap Hausman test can be conducted, using an adaptation of the bootstrap Hausman test example in section [12.4.6](#).

Simpler is to test  $H_0: \gamma = 0$  in the auxiliary OLS regression

$$y_{it} = \mathbf{x}'_{it}\beta + \bar{\mathbf{x}}'_{1i}\gamma + v_{it}$$

where  $\mathbf{x}_1$  denotes only time-varying regressors. A Wald test of  $\gamma = 0$  can be shown to be asymptotically equivalent to the standard test when the RE

estimator is fully efficient under  $H_0$ . A summary of related tests for FE versus RE is given in [Baltagi \(2021, 85–92\)](#).

In the more likely case that the RE estimator is not fully efficient, [Wooldridge \(2010, 332–323\)](#) proposes performing the Wald test using cluster-robust standard errors. The individual averages  $\bar{x}_{1i}$  have already been created for the correlated in section [8.7.4](#). We have

```
. * Cluster-robust Hausman test using method of Wooldridge (2010)
. qui regress lwage $xlist meanexp meanexp2 meanwks, vce(cluster id)
. test meanexp meanexp2 meanwks
( 1) meanexp = 0
( 2) meanexp2 = 0
( 3) meanwks = 0
F(  3,    594) =   597.47
Prob > F =      0.0000
```

The test strongly rejects the null hypothesis, and we conclude that the RE model is not appropriate.

The community-contributed command `xtoverid` ([Schaffer and Stillman 2006](#)) implements the preceding test following command `xtreg, re vce(robust)`. We have

```
. * Cluster-robust Hausman test using xtovierid command
. qui xtreg lwage $xlist, re vce(robust)
. xtoverid
Test of overidentifying restrictions: fixed vs random effects
Cross-section time-series model: xtreg re robust cluster(id)
Sargan-Hansen statistic 1792.412 Chi-sq(3) P-value = 0.0000
```

This presents a  $\chi^2(q)$  version of the test that equals  $q$  times the preceding  $F$ -test statistic; here  $3 \times 597.47 = 1792.41$ .

## 8.8.6 Prediction

The `predict` postestimation command after `xtreg` provides estimated residuals and fitted values following estimation of the individual-effects model  $y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + \varepsilon_{it}$ .

The estimated individual-specific error  $\hat{\alpha}_i = \bar{y}_i - \bar{\mathbf{x}}_{it}'\hat{\beta}$  is obtained by using the `u` option; the estimated idiosyncratic error  $\hat{\varepsilon}_{it} = y_{it} - \hat{\alpha}_i - \mathbf{x}_{it}'\hat{\beta}$  is obtained by using the `e` option; and the `ue` option gives  $\hat{\alpha}_i + \hat{\varepsilon}_{it}$ .

Fitted values of the dependent variable differ according to whether the estimated individual-specific error is used. The fitted value  $y_{it} = \hat{\alpha} + \mathbf{x}_{it}'\hat{\beta}$ , where  $\hat{\alpha} = N^{-1} \sum_i \hat{\alpha}_i$ , is obtained by using the `xb` option. The fitted value  $y_{it} = \hat{\alpha}_i + \mathbf{x}_{it}'\hat{\beta}$  is obtained by using the `xbu` option.

As an example, we contrast OLS and RE in-sample fitted values.

```
. * Prediction after OLS and RE estimation
. qui regress lwage exp exp2 wks ed, vce(cluster id)
. predict xbols, xb
. qui xtreg lwage exp exp2 wks ed, re
. predict xbre, xb
. predict xbure, xbu
. summarize lwage xbols xbre xbure
```

Variable	Obs	Mean	Std. dev.	Min	Max
lwage	4,165	6.676346	.4615122	4.60517	8.537
xbols	4,165	6.676346	.2457572	5.850037	7.200861
xbre	4,165	6.676346	.6205324	5.028067	8.22958
xbure	4,165	6.676346	.4082951	5.29993	7.968179

```
. correlate lwage xbols xbre xbure
(obs=4,165)
```

	lwage	xbols	xbre	xbure
lwage	1.0000			
xbols	0.5325	1.0000		
xbre	0.4278	0.8034	1.0000	
xbure	0.9375	0.6019	0.4836	1.0000

The RE prediction  $\hat{\alpha} + \mathbf{x}_{it}'\hat{\beta}$  is not as highly correlated with `lwage` as is the OLS prediction (0.43 versus 0.53), which was expected because the OLS estimator maximizes this correlation.

When instead we use  $\hat{\alpha}_i + \mathbf{x}_{it}'\hat{\beta}$  so the fitted individual effect is included, the correlation of the prediction with `lwage` increases greatly to 0.94. In a short panel, however, these predictions are not consistent, because

each individual prediction  $\widehat{\alpha}_i = \bar{y}_i - \bar{\mathbf{x}}'_{it} \widehat{\boldsymbol{\beta}}$  is based on only  $T$  observations and  $T \not\rightarrow \infty$ .

## 8.9 First-difference estimator

Consistent estimation of  $\beta$  in the FE model requires eliminating the  $\alpha_i$ . One way to do so is to mean-difference, yielding the within estimator. An alternative way is to FD, leading to the FD estimator.

For analysis of static models, the FE estimator is traditionally favored for two reasons. First, in a balanced panel, it is generally more efficient because it is the efficient estimator if the  $\varepsilon_{it}$  are i.i.d., whereas the FD estimator is efficient under the less realistic assumption that  $(\varepsilon_{it} - \varepsilon_{i,t-1})$  is i.i.d. Second, with unbalanced data more observations are lost using the FD estimator. For example, if  $T = 4$  and for individual  $i$  only observations 1 and 3 are available, then we cannot compute  $(y_{it} - y_{i,t-1})$  for any  $t$ , but we can compute  $(y_{i3} - \bar{y}_i)$ , where  $\bar{y}_i = (y_{i1} + y_{i3})/2$ .

The FD estimator has the advantage of relying on weaker exogeneity assumptions, explained below, that become important in dynamic models, with lagged values of  $y_{it}$  as regressor, presented in the next chapter.

### 8.9.1 FD estimator

The FD estimator is obtained by performing OLS on the first-differenced variables

$$(y_{it} - y_{i,t-1}) = (\mathbf{x}_{it} - \mathbf{x}_{i,t-1})'\beta + (\varepsilon_{it} - \varepsilon_{i,t-1}) \quad (8.12)$$

First-differencing has eliminated  $\alpha_i$ , so OLS estimation of this model leads to consistent estimates of  $\beta$  in the FE model. The coefficients of time-invariant regressors are not identified, because then  $x_{it} - x_{i,t-1} = 0$ , as was the case for the within estimator.

The FD estimator is not provided as an option to `xtreg`. Instead, the estimator can be computed by using `regress` and Stata time-series operators to compute the FDs. We have

```

. sort id t
. * FD estimator with cluster--robust standard errors
. regress D.(lwage exp exp2 wks ed), vce(cluster id)
note: D.exp omitted because of collinearity.
note: D.ed omitted because of collinearity.

Linear regression                                         Number of obs      =      3,570
                                                F(2, 594)        =     22.66
                                                Prob > F       =     0.0000
                                                R-squared       =     0.0041
                                                Root MSE        =     .18156
                                                (Std. err. adjusted for 595 clusters in id)

```

D.lwage	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
exp D1.	0 (omitted)				
exp2 D1.	-.0005321	.0000808	-6.58	0.000	-.0006908 -.0003734
wks D1.	-.0002683	.0011783	-0.23	0.820	-.0025824 .0020459
ed D1.	0 (omitted)				
_cons	.1170654	.0040974	28.57	0.000	.1090182 .1251126

As expected, the coefficient for education is not identified, because `ed` here is time invariant. The coefficient for `wks` actually changes sign compared with the other estimators, though it is highly statistically insignificant.

The FD estimator, like the within estimator, provides consistent estimators when the individual effects are fixed. For panels with  $T = 2$ , the FD and within estimators are equivalent; otherwise, the two differ. For static models, the FE model is used because it is the efficient estimator if the idiosyncratic error  $\varepsilon_{it}$  is i.i.d.

The FD estimator seemingly uses one less year of data compared with the within estimator because the FD output lists 3,570 observations rather than 4,165. This, however, is misleading. Using the LSDV interpretation of the within estimator, the within estimator essentially loses 595 observations by estimating the  $T$  fixed effects  $\alpha_1, \dots, \alpha_T$ .

The `xtivreg` command presented in section 9.2.2 has an option for FD estimation. For the current example, the command `xtivreg2 lwage exp exp2 wks ed, fd cluster(id) small` yields exactly the same results.

### 8.9.2 Strict and weak exogeneity

From (8.6), the within estimator requires that  $\varepsilon_{it} - \bar{\varepsilon}_i$  be uncorrelated with  $\mathbf{x}_{it} - \bar{\mathbf{x}}_i$ . This is the case under the assumption of strict exogeneity or strong exogeneity that

$$E(\varepsilon_{it} | \alpha_i, \mathbf{x}_{i1}, \dots, \mathbf{x}_{it}, \dots, \mathbf{x}_{iT}) = 0$$

From (8.12), the FD estimator requires that  $\varepsilon_{it} - \varepsilon_{i,t-1}$  be uncorrelated with  $\mathbf{x}_{it} - \mathbf{x}_{i,t-1}$ . This is the case under the assumption of weak or sequential exogeneity that

$$E(\varepsilon_{it} | \alpha_i, \mathbf{x}_{i1}, \dots, \mathbf{x}_{it}) = 0$$

This is a considerably weaker assumption because it permits future values of the regressors to be correlated with the error, as will be the case if the regressor is a lagged dependent variable.

As long as there is no feedback from the idiosyncratic shock today to a covariate tomorrow, this distinction is unnecessary when fitting static models. It becomes important for dynamic models (see section 9.4) because then strict exogeneity no longer holds and we turn to the FD estimator.

## 8.10 Panel-data management

Stata `xt` commands require panel data to be in long form, which means that each individual–time pair is a separate observation. Some datasets instead store panel data in wide form, which has the advantage of using less space. Sometimes, the observational unit is the individual, and a single observation has all time periods for that individual. And sometimes the observational unit is a time period, and a single observation has all individuals for that time period.

We illustrate how to move from wide form to long form and vice versa by using the `reshape` command. Our example is for panel data, but `reshape` can also be used in other contexts where data are grouped, such as clustered data grouped by village rather than panel data grouped by time.

For data management commands for unbalanced samples, see section 19.11.

### 8.10.1 Wide-form data

We consider a small dataset that is originally in wide form, with each observation containing all years of data for an individual. The dataset is a subset of the data described in section 9.5.1. Each observation is a state and has all years of data for that state. We have

```
. * Wide-form data (observation is a state)
. qui use mus208cigarwide, clear
. list, clean
```

	state	lnp63	lnc63	lnp64	lnc64	lnp65	lnc65
1.	1	4.5	4.5	4.6	4.6	4.5	4.6
2.	2	4.4	4.8	4.3	4.8	4.3	4.8
3.	3	4.5	4.6	4.5	4.6	4.5	4.6
4.	4	4.4	5.0	4.4	4.9	4.4	4.9
5.	5	4.5	5.1	4.5	5.0	4.5	5.0
6.	6	4.5	5.1	4.5	5.1	4.5	5.1
7.	7	4.3	5.5	4.3	5.5	4.3	5.5
8.	8	4.5	4.9	4.6	4.8	4.5	4.9
9.	9	4.5	4.7	4.5	4.7	4.6	4.6
10.	10	4.5	4.6	4.6	4.5	4.5	4.6

The data contain a state identifier, `state`; three years of data on log price, `lnp63–lnp65`; and three years of data on log sales, `lnc63–lnc65`. The data are for 10 states.

### 8.10.2 Convert wide form to long form

The data can be converted from wide form to long form by using `reshape long`. The desired dataset will have an observation as a state–year pair. The variables should be a state identifier, a year identifier, and the current state–year observations on `lnp` and `lnc`.

The simple command `reshape long` actually does this automatically because it interprets the suffixes 63–65 as denoting the grouping that needs to be expanded to long form. We use a more detailed version of the command that spells out exactly what we want to do and leads to exactly the same result as `reshape long` without arguments. We have

```
. * Convert from wide form to long form (observation is a state-year pair)
. reshape long lnp lnc, i(state) j(year)
(j = 63 64 65)
```

Data	Wide	->	Long
Number of observations	10	->	30
Number of variables	7	->	4
j variable (3 values)		->	year
xij variables:			
	lnp63 lnp64 lnp65	->	lnp
	lnc63 lnc64 lnc65	->	lnc

The output indicates that we have expanded the dataset from 10 observations (10 states) to 30 observations (30 state–year pairs). A year-identifier variable, `year`, has been created. The wide-form data `lnp63–lnp65` have been collapsed to `lnp` in long form, and `lnc63–lnc65` have been collapsed to `lnc`.

We now list the first six observations of the new long-form data.

```
. * Long-form data (observation is a state)
. list in 1/6, sepby(state)
```

	state	year	lnp	lnc
1.	1	63	4.5	4.5
2.		64	4.6	4.6
3.		65	4.5	4.6
4.	2	63	4.4	4.8
5.		64	4.3	4.8
6.		65	4.3	4.8

Any year-invariant variables will also be included in the long-form data. Here the state-identifier variable, `state`, is the only such variable.

### 8.10.3 Convert long form to wide form

Going the other way, data can be converted from long form to wide form by using `reshape wide`. The desired dataset will have an observation as a state. The constructed variables should be a state identifier and observations on `lnp` and `lnc` for each of the three years 63–65.

The `reshape wide` command without arguments actually does this automatically because it interprets `year` as the relevant time identifier and adds suffixes 63–65 to the variables `lnp` and `lnc`, which are varying with `year`. We use a more detailed version of the command that spells out exactly what we want to do and leads to exactly the same result. We have

```
. * Reconvert from long form to wide form (observation is a state)
. reshape wide lnp lnc, i(state) j(year)
(j = 63 64 65)
```

Data	Long	->	Wide
Number of observations	30	->	10
Number of variables	4	->	7
j variable (3 values)	year	->	(dropped)
xij variables:			
	lnp	->	lnp63 lnp64 lnp65
	lnc	->	lnc63 lnc64 lnc65

The output indicates that we have collapsed the dataset from 30 observations (30 state–year pairs) to 10 observations (10 states). The `year` variable has been dropped. The long-form data `lnp` has been expanded to `lnp63–lnp65` in wide form, and `lnc` has been expanded to `lnc63–lnc65`.

A complete listing of the wide-form dataset is

	state	lnp63	lnc63	lnp64	lnc64	lnp65	lnc65
1.	1	4.5	4.5	4.6	4.6	4.5	4.6
2.	2	4.4	4.8	4.3	4.8	4.3	4.8
3.	3	4.5	4.6	4.5	4.6	4.5	4.6
4.	4	4.4	5.0	4.4	4.9	4.4	4.9
5.	5	4.5	5.1	4.5	5.0	4.5	5.0
6.	6	4.5	5.1	4.5	5.1	4.5	5.1
7.	7	4.3	5.5	4.3	5.5	4.3	5.5
8.	8	4.5	4.9	4.6	4.8	4.5	4.9
9.	9	4.5	4.7	4.5	4.7	4.6	4.6
10.	10	4.5	4.6	4.6	4.5	4.5	4.6

This is exactly the same as the original `mus208cigarwide.dta` dataset, listed in section [8.10.1](#).

#### 8.10.4 An alternative wide-form data

The wide form we considered had each state as the unit of observation. An alternative is that each year is the observation. Then the preceding commands are reversed so that we have `i(year)` `j(state)` rather than `i(state)` `j(year)`.

To demonstrate this case, we first need to create the data in wide form with `year` as the observational unit. We do so by converting the current data, in wide form with `state` as the observational unit, to long form with 30 observations as presented above and then use `reshape wide` to create wide-form data with `year` as the observational unit.

```

. * Create alternative wide-form data (observation is a year)
. qui reshape long lnp lnc, i(state) j(year)
. reshape wide lnp lnc, i(year) j(state)
(j = 1 2 3 4 5 6 7 8 9 10)

Data                                Long    ->    Wide

```

---

Number of observations	30	->	3
Number of variables	4	->	21
j variable (10 values)	state	->	(dropped)
xij variables:		->	lnp1 lnp2 ... lnp10
		->	lnc1 lnc2 ... lnc10

---

```

. list year lnp1 lnp2 lnc1 lnc2, clean
      year    lnp1    lnp2    lnc1    lnc2
1.      63     4.5     4.4     4.5     4.8
2.      64     4.6     4.3     4.6     4.8
3.      65     4.5     4.3     4.6     4.8

```

The wide form has 3 observations (1 per year) and 21 variables (`lnp` and `lnc` for each of 10 states plus `year`).

We now have data in wide form with `year` as the observational unit. To use `xt` commands, we use `reshape long` to convert to long-form data with an observation for each state–year pair. We have

```

. * Convert from wide form (observation is year) to long form (year-state)
. reshape long lnp lnc, i(year) j(state)
(j = 1 2 3 4 5 6 7 8 9 10)

```

```

Data                                Wide    ->    Long

```

---

Number of observations	3	->	30
Number of variables	21	->	4
j variable (10 values)		->	state
xij variables:	lnp1 lnp2 ... lnp10	->	lnp
	lnc1 lnc2 ... lnc10	->	lnc

---

```

. list in 1/6, clean
      year    state    lnp    lnc
1.      63        1    4.5    4.5
2.      63        2    4.4    4.8
3.      63        3    4.5    4.6
4.      63        4    4.4    5.0
5.      63        5    4.5    5.1
6.      63        6    4.5    5.1

```

The data are now in long form, as in section [8.10.2](#).

## 8.11 Additional resources

FE and RE estimators appear in many econometrics texts; [Wooldridge \(2010\)](#) in particular has a very extensive coverage. Standard panel texts are [Baltagi \(2021\)](#) and [Hsiao \(2014\)](#). The key Stata reference is [XT] *Stata Longitudinal/Panel-Data Reference Manual*, especially [XT] **xt** and [XT] **xtreg**. Useful online `help` categories include `xt` and `xtreg`.

## 8.12 Exercises

1. For the data of section [8.3](#), use `xtsum` to describe the variation in `occ`, `smsa`, `ind`, `ms`, `union`, `fem`, and `blk`. Which of these variables are time invariant? Use `xttab` and `xttrans` to provide interpretations of how `occ` changes for individuals over the seven years. Provide a time-series plot of `exp` for the first 10 observations, and provide interpretation. Provide a scatterplot of `lwage` against `ed`. Is this plot showing within variation, between variation, or both?
2. For the data of section [8.3](#), manually obtain the three standard deviations of `lwage` given by the `xtsum` command. For the overall standard deviation, use `summarize`. For the between standard deviation, compute by `id`: `egen meanwage = mean(lwage)`, and apply `summarize to (meanwage - grandmean)` for `t==1`, where *grandmean* is the grand mean over all observations. For the within standard deviation, apply `summarize to (lwage - meanwage)`. Compare your standard deviations with those from `xtsum`. Does  $s_O^2 \approx s_W^2 + s_B^2$ ?
3. For the model and data of section [8.4](#), compare PGLS estimators under the following assumptions about the error process: independent, exchangeable, AR(2), and MA(6). Also, compare the associated standard-error estimates obtained by using default standard errors and by using cluster-robust standard errors. You will find it easiest if you combine results using `estimates table`. What happens if you try to fit the model with no structure placed on the error correlations?
4. For the model and data of section [8.5](#), obtain the within estimator by applying `regress` to [\(8.7\)](#). Hint: For example, for variable *x*, type by `id`: `egen avex = mean(x)` followed by `summarize x` and then `generate mdx = x - avex + r(mean)`. Verify that you get the same estimated coefficients as you would with `xtreg, fe`.
5. For the model and data of section [8.6](#), compare the RE estimators that were obtained by using `xtreg` with the `re`, `mle`, and `pa` options and `xtgee` with the `corr(exchangeable)` option. Also, compare the associated standard-error estimates obtained by using default standard errors and by using cluster-robust standard errors. You will find it easiest if you combine results using `estimates table`.

6. Consider the RE model output given in section [8.7](#). Verify that, given the estimated values of `e_sigma` and `u_sigma`, application of the formulas in that section leads to the estimated values of `rho` and `theta`.



# **Chapter 9**

## **Linear panel-data models: Extensions**

## 9.1 Introduction

The essential panel methods for linear models, most notably, the important distinction between fixed-effects (FE) and random-effects (RE) models, were presented in chapter [8](#).

In this chapter, we present other panel methods for the linear model. For short panels, we consider instrumental-variables (IV) estimation and estimation of dynamic models with lagged dependent variables as regressors. We also provide a brief discussion of estimation of long panels. Nonlinear panel models are presented in chapter 22.

## 9.2 Panel instrumental-variables estimation

IV methods have been extended from cross-sectional data (see chapter 7 for an explanation) to panel data. Estimation still needs to eliminate the  $\alpha_i$ , if the FE model is appropriate, and inference needs to control for the clustering inherent in panel data.

In this section, we detail `xtivreg`, which is a panel extension of the cross-sectional command `ivregress`. The subsequent two sections present more specialized IV estimators and commands that are applicable in situations where regressors from periods other than the current period are used as instruments.

### 9.2.1 Panel IV

If a pooled model is appropriate with  $y_{it} = \alpha + \mathbf{x}'_{it}\beta + u_{it}$  and instruments  $\mathbf{z}_{it}$  exist satisfying  $E(u_{it}|\mathbf{z}_{it}) = 0$ , then consistent estimation is possible by two-stage least-squares (2SLS) regression of  $y_{it}$  on  $\mathbf{x}_{it}$  with instruments  $\mathbf{z}_{it}$ . The `ivregress` command can be used, with subsequent statistical inference based on cluster-robust standard errors.

More often, we use the individual-effects model

$$y_{it} = \mathbf{x}'_{it}\beta + \alpha_i + \varepsilon_{it}, \quad t = 1, \dots, T_i, \quad i = 1, \dots, N \quad (9.1)$$

which has two error components,  $\alpha_i$  and  $\varepsilon_{it}$ . The FE and first-difference (FD) estimators provide consistent estimates of the coefficients of the time-varying regressors under a limited form of endogeneity of the regressors— $\mathbf{x}_{it}$  may be correlated with the fixed effects  $\alpha_i$  but not with  $\varepsilon_{it}$ .

We now consider a richer type of endogeneity, with  $\mathbf{x}_{it}$  correlated with  $\varepsilon_{it}$ . We need to assume the existence of instruments  $\mathbf{z}_{it}$  that are correlated with  $\mathbf{x}_{it}$  and uncorrelated with  $\varepsilon_{it}$ . The panel IV procedure is to suitably transform the individual-effects model to control for  $\alpha_i$  and then apply IV to the transformed model.

## 9.2.2 The xtivreg command

The `xtivreg` command implements 2SLS regression with options corresponding to those for the `xtreg` command. The syntax is similar to that for the cross-sectional `ivregress` command:

```
xtivreg depvar [ varlist1 ] (varlist2=varlist_iv) [ if ] [ in ] [ , options ]
```

The four main options are `fe`, `fd`, `re`, and `be`. The `fe` option performs within 2SLS regression of  $y_{it} - \bar{y}_i$  on an intercept and  $\mathbf{x}_{it} - \bar{\mathbf{x}}_i$  with the instruments  $\mathbf{z}_{it} - \bar{\mathbf{z}}_i$ . The `fd` option, not available in `xtreg`, performs FD 2SLS regression of  $y_{it} - y_{i,t-1}$  on an intercept and  $\mathbf{x}_{it} - \mathbf{x}_{i,t-1}$  with the instruments  $\mathbf{z}_{it} - \mathbf{z}_{i,t-1}$ . The `re` option performs RE 2SLS regression of  $y_{it} - \hat{\theta}_i \bar{y}_i$  on an intercept and  $\mathbf{x}_{it} - \hat{\theta}_i \bar{\mathbf{x}}_i$  with the instruments  $\mathbf{z}_{it} - \hat{\theta}_i \bar{\mathbf{z}}_i$ , and the additional options `ec2sls` and `nosa` provide variations of this estimator. The `be` option performs between 2SLS regression of  $\bar{y}_i$  on  $\bar{\mathbf{x}}_i$  with instruments  $\bar{\mathbf{z}}_i$ . Other options include `first` to report first-stage regression results and `regress` to ignore the instruments and instead estimate the parameters of the transformed model by ordinary least squares (OLS). The `vce()` options include heteroskedastic-robust and cluster-robust standard errors.

The community-contributed `xtivreg2` command ([Schaffer 2005](#)) provides a broader range of IV-related estimators, including 2SLS, GMM, LIML,  $k$ -class and continuous updating, for FE and FD models. The command includes tests of weak instruments and overidentifying restrictions.

As usual, exogenous regressors are instrumented by themselves. For endogenous regressors, we can proceed as in the cross-sectional case, obtaining an additional variable that does not directly determine  $y_{it}$  but is correlated with the variable being instrumented. In the simplest case, the instrument is an external instrument, a variable that does not appear directly as a regressor in the model. This is the same IV identification strategy as that used with cross-sectional data.

## 9.2.3 Application of the xtivreg command

Consider the section 8.3 example of regression of `lwage` on `exp`, `exp2`, `wks`, and `ed`. We assume the experience variables `exp` and `exp2` are exogenous and that `ed` is correlated with the time-invariant component of the error but is uncorrelated with the time-varying component of the error. Given just these assumptions, we need to control for fixed effects. From section 8.6, the within estimator yields consistent estimates of coefficients of `exp`, `exp2`, and `wks`, whereas the coefficient of `ed` is not identified because it is a time-invariant regressor.

Now suppose that the regressor `wks` is correlated with the time-varying component of the error. Then the within estimator becomes inconsistent, and we need to instrument for `wks`. We suppose that `ms` (marital status) is a suitable instrument. This requires an assumption that marital status does not directly determine the wage rate but is correlated with weeks worked. Because the effects here are fixed, the `fe` or `fd` option of `xtivreg` needs to be used.

Formally, we have assumed that the instruments—here `exp`, `exp2`, and `ms`—satisfy the strong exogeneity assumption that

$$E(\varepsilon_{it} | \alpha_i, \mathbf{z}_{i1}, \dots, \mathbf{z}_{it}, \dots, \mathbf{z}_{iT}) = 0$$

so that instruments and errors are uncorrelated in all periods. One consequence of this strong assumption is that panel IV estimators are consistent even if the  $\varepsilon_{it}$  are serially correlated, so cluster-robust standard errors could be used.

We use `xtivreg` with the `fe` option to eliminate the fixed effects. We drop the unidentified time-invariant regressor `ed`—the same results are obtained if it is included. We obtain

```

. * Panel IV example: FE with wks instrumented by external instrument ms
. qui use mus208psid
. xtivreg lwage exp exp2 (wks = ms), fe vce(robust)

Fixed-effects (within) IV regression
Group variable: id
R-squared:
    Within = .
    Between = 0.0172
    Overall = 0.0284

Number of obs      =      4,165
Number of groups  =       595
Obs per group:
    min =          7
    avg =        7.0
    max =          7

Wald chi2(3)      =     12295.89
corr(u_i, Xb) = -0.8499
Prob > chi2       =      0.0000
(Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
wks	-.1149742	.3707276	-0.31	0.756	-.8415868 .6116385
exp	.1408101	.0902921	1.56	0.119	-.0361591 .3177794
exp2	-.0011207	.0022925	-0.49	0.625	-.0056139 .0033726
_cons	9.83932	16.75004	0.59	0.557	-22.99015 42.66879
sigma_u	1.0980369				
sigma_e	.51515503				
rho	.81959748	(fraction of variance due to u_i)			

Instrumented: wks

Instruments: exp exp2 ms

The estimates imply that, surprisingly, wages decrease by 11.5% for each additional week worked, though the coefficient is statistically insignificant. Wages increase with experience until a peak at 64 years [ $= 0.1408/(2 \times 0.0011)$ ].

Comparing the IV results with those given in section [8.5.3](#) using `xtreg, fe`, we see the coefficient of the endogenous variable `wks` has changed sign and is many times larger in absolute value, whereas the coefficients of the exogenous experience regressors are less affected. For these data, the IV standard errors are more than 10 times larger. Because the instrument `ms` is not very correlated with `wks`, IV regression leads to a substantial loss in estimator efficiency.

## 9.2.4 Panel IV extensions

We used the external instrument  $w_{ks}$  as the instrument for  $w_{ks}$ .

An alternative is to use  $w_{ks}$  from a period other than the current period as the instrument. This has the attraction of being reasonably highly correlated with the variable being instrumented, but it is not necessarily a valid instrument. In the simplest panel model  $y_{it} = \mathbf{x}'_{it}\beta + \varepsilon_{it}$ , if the errors  $\varepsilon_{it}$  are independent, then any variable in any period that is not in  $\mathbf{x}_{it}$  is a valid instrument. Once we introduce an individual effect as in (9.1) and mean-transform the model, more care is needed.

The next two sections present, respectively, the Hausman–Taylor estimator and the Arellano–Bond estimator, which use as instruments regressors from periods other than the current period. In some settings, a Bartik or shift-share instrument can be used; see [Goldsmith-Pinkham, Sorkin, and Swift \(2020\)](#).

## 9.3 Hausman–Taylor estimator

We consider the FE model. The FE and FD estimators provide consistent estimators but not for the coefficients of time-invariant regressors because these are then not identified. The Hausman–Taylor estimator is an IV estimator that additionally enables the coefficients of time-invariant regressors to be estimated. It does so by making the stronger assumption that some specified regressors are uncorrelated with the fixed effects. Then values of these regressors in periods other than the current period can be used as instruments.

### 9.3.1 Hausman–Taylor estimator

The key step is to distinguish between regressors uncorrelated with the fixed effects and those potentially correlated with the fixed effects. The method additionally distinguishes between time-varying and time-invariant regressors.

The individual-effects model is then written as

$$y_{it} = \mathbf{x}'_{1it}\boldsymbol{\beta}_1 + \mathbf{x}'_{2it}\boldsymbol{\beta}_2 + \mathbf{w}'_{1i}\boldsymbol{\gamma}_1 + \mathbf{w}'_{2i}\boldsymbol{\gamma}_2 + \alpha_i + \varepsilon_{it} \quad (9.2)$$

where regressors with subscript 1 are specified to be uncorrelated with  $\alpha_i$  and regressors with subscript 2 are specified to be correlated with  $\alpha_i$ ,  $\mathbf{w}$  denotes time-invariant regressors, and  $\mathbf{x}$  now denotes time-varying regressors. All regressors are assumed to be uncorrelated with  $\varepsilon_{it}$ , whereas `xtivreg` explicitly deals with such correlation.

The Hausman–Taylor method is based on the RE transformation that leads to the model

$$\tilde{y}_{it} = \tilde{\mathbf{x}}'_{1it}\boldsymbol{\beta}_1 + \tilde{\mathbf{x}}'_{2it}\boldsymbol{\beta}_2 + \tilde{\mathbf{w}}'_{1i}\boldsymbol{\gamma}_1 + \tilde{\mathbf{w}}'_{2i}\boldsymbol{\gamma}_2 + \tilde{\alpha}_i + \tilde{\varepsilon}_{it}$$

where, for example,  $\tilde{\mathbf{x}}_{1it} = \mathbf{x}_{1it} - \hat{\theta}_i \bar{\mathbf{x}}_{1i}$ , and the formula for  $\hat{\theta}_i$  is given in [XT] **xhtaylor**.

The RE transformation is used because, unlike the within transform, here  $\tilde{\mathbf{w}}_{1i} \neq 0$  and  $\tilde{\mathbf{w}}_{2i} \neq 0$ , so  $\gamma_1$  and  $\gamma_2$  can be estimated. But  $\tilde{\alpha}_i = \alpha_i(1 - \hat{\theta}_i) \neq 0$ , so the fixed effect has not been eliminated, and  $\tilde{\alpha}_i$  is correlated with  $\tilde{\mathbf{x}}_{2it}$  and with  $\tilde{\mathbf{w}}_{2i}$ . This correlation is dealt with by IV estimation. For  $\tilde{\mathbf{x}}_{2it}$ , the instrument used is  $\ddot{\mathbf{x}}_{2it} = \mathbf{x}_{2it} - \bar{\mathbf{x}}_{2i}$ , which can be shown to be uncorrelated with  $\alpha_i$ . For  $\tilde{\mathbf{w}}_{2i}$ , the instrument is  $\bar{\mathbf{x}}_{1i}$ , so the method requires that the number of time-varying exogenous regressors be at least as large as the number of time-invariant endogenous regressors. The method uses  $\ddot{\mathbf{x}}_{1it}$  as an instrument for  $\tilde{\mathbf{x}}_{1it}$  and  $\mathbf{w}_{1i}$  as an instrument for  $\tilde{\mathbf{w}}_{1i}$ . Essentially,  $\mathbf{x}_1$  is used as an instrument twice: as  $\ddot{\mathbf{x}}_{1it}$  and as  $\bar{\mathbf{x}}_{1i}$ . By using the average of  $\bar{\mathbf{x}}_{1i}$  in forming instruments, we are using data from other periods to form instruments.

### 9.3.2 The xhtaylor command

The `xhtaylor` command performs IV estimation of the parameters of (9.2) using the instruments  $\ddot{\mathbf{x}}_{1it}$ ,  $\ddot{\mathbf{x}}_{2it}$ ,  $\mathbf{w}_{1i}$ , and  $\bar{\mathbf{x}}_{1i}$ . The syntax of the command is

```
xhtaylor depvar indepvars [if] [in] [weight], endog(varlist) [options]
```

Here all the regressors are given in *indepvars*, and the subset of these that are potentially correlated with  $\alpha_i$  are given in `endog(varlist)`. The `xhtaylor` command provides an option to compute cluster-robust standard errors.

The options include `amacurdy`, which uses a wider range of instruments. Specifically, the Hausman–Taylor method requires that  $\bar{\mathbf{x}}_{1i}$  be uncorrelated with  $\alpha_i$ . If each  $\mathbf{x}_{1it}$ ,  $t = 1, \dots, T$ , is uncorrelated with  $\alpha_i$ , then more instruments are available, and we can use as instruments  $\ddot{\mathbf{x}}_{1it}$ ,  $\ddot{\mathbf{x}}_{2it}$ ,  $\mathbf{w}_{1i}$ , and  $\mathbf{x}_{1i1}, \dots, \mathbf{x}_{1iT}$ .

### 9.3.3 Application of the xhtaylor command

The dataset used here, attributed to [Baltagi and Khanti-Akom \(1990\)](#) and [Cornwell and Rupert \(1988\)](#), was originally applied to the Hausman–Taylor estimator. We reproduce that application here. It uses a wider set of regressors than we have used to this point.

The goal is to obtain a consistent estimate of the coefficient  $ed$  because there is great interest in the impact of education on wages. Education is clearly endogenous. It is assumed that education is correlated only with the individual-specific component of the error  $\alpha_i$ . In principle, within estimation gives a consistent estimator, but in practice, no estimator is obtained because  $ed$  is time invariant, so its coefficient cannot be estimated.

The Hausman–Taylor estimator is used instead, assuming that only a subset of the regressors is correlated with  $\alpha_i$ . Identification requires that there be at least one time-varying regressor that is uncorrelated with the fixed effects. [Cornwell and Rupert \(1988\)](#) assumed that, for the time-varying regressors, `exp`, `exp2`, `wks`, `ms`, and `union` were endogenous, whereas `occ`, `south`, `smsa`, and `ind` were exogenous. And for the time-invariant regressors, `ed` is endogenous and `fem` and `blk` are exogenous. The `xthtaylor` command requires distinction only between endogenous and exogenous regressors because it can determine which regressors are time varying and which are not.

We obtain

```

. * Hausman-Taylor example of Baltagi and Khanti-Akom (1990)
. xtaylor lwage occ south smsa ind exp exp2 wks ms union fem blk ed,
> endog(exp exp2 wks ms union ed) vce(robust)

Hausman-Taylor estimation
Group variable: id
Number of obs      =      4,165
Number of groups  =       595
Obs per group:
min =           7
avg =           7
max =           7

Random effects u_i ~ i.i.d.          Wald chi2(12)      =   33204.76
                                                Prob > chi2     =    0.0000
                                                (Std. err. adjusted for 595 clusters in id)

```

lwage	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
TVexogenous						
occ	-.0207047	.0190066	-1.09	0.276	-.0579569	.0165475
south	.0074398	.0785588	0.09	0.925	-.1465327	.1614123
smsa	-.0418334	.0285698	-1.46	0.143	-.0978291	.0141624
ind	.0136039	.0222459	0.61	0.541	-.0299973	.0572051
TVendogenous						
exp	.1131328	.0040568	27.89	0.000	.1051816	.1210839
exp2	-.0004189	.0000823	-5.09	0.000	-.0005803	-.0002575
wks	.0008374	.0008666	0.97	0.334	-.0008611	.0025359
ms	-.0298508	.0268188	-1.11	0.266	-.0824147	.0227132
union	.0327714	.0250575	1.31	0.191	-.0163404	.0818832
TIexogenous						
fem	-.1309236	.11753	-1.11	0.265	-.3612781	.0994309
blk	-.2857479	.1704617	-1.68	0.094	-.6198467	.0483509
TIendogenous						
ed	.137944	.0216481	6.37	0.000	.0955145	.1803734
_cons	2.912726	.307723	9.47	0.000	2.3096	3.515852
sigma_u	.94180304					
sigma_e	.15180273					
rho	.97467788	(fraction of variance due to u_i)				

Note: TV refers to time varying; TI refers to time invariant.

Compared with the RE estimates given in section 8.7, the coefficient of `ed` has increased from 0.112 to 0.138, and the standard error has increased from 0.0084 to 0.0216.

For the regular IV estimator to be consistent, it is necessary to argue that any instruments are uncorrelated with the error term. Similarly, for the

Hausman–Taylor estimator to be consistent, it is necessary to argue that all regressors are uncorrelated with the idiosyncratic error  $\varepsilon_{it}$  and that a specified subset of the regressors is uncorrelated with the fixed effects  $\alpha_i$ . This strong assumption can be tested by the community-contributed command `xtoverid` following command `xttaylor`.

## 9.4 Arellano–Bond estimator

With panel data, the dependent variable is observed over time, opening up the possibility of estimating parameters of dynamic models that specify the dependent variable for an individual to depend in part on its values in previous periods. As in the nonpanel case, however, care is needed because OLS with a lagged dependent variable and serially correlated error leads to inconsistent parameter estimates.

We consider estimation of FE models for short panels when one or more lags of the dependent variable are included as regressors. Then the fixed effect needs to be eliminated by first differencing rather than mean differencing for reasons given at the end of section 9.4.1. Consistent estimators can be obtained by IV estimation of the parameters in the FD model, using appropriate lags of regressors as the instruments. This estimator, called the Arellano–Bond estimator, can be performed, with some considerable manipulation, by using the iv command `ivregress` or `xtivreg`. But it is much easier to use the specialized commands `xtabond`, `xtdpdsys`, and `xtdpd`. These commands also enable more efficient estimation and provide appropriate model specification tests.

### 9.4.1 Dynamic model

The general model considered is an autoregressive model of order  $p$  in  $y_{it}$  [an AR( $p$ ) model] with  $y_{i,t-1}, \dots, y_{i,t-p}$  as regressors, as well as the regressors  $\mathbf{x}_{it}$ . The model is

$$y_{it} = \gamma_1 y_{i,t-1} + \dots + \gamma_p y_{i,t-p} + \mathbf{x}'_{it} \boldsymbol{\beta} + \alpha_i + \varepsilon_{it}, \quad t = p+1, \dots, T \quad (9.3)$$

where  $\alpha_i$  is a fixed effect. The regressors  $\mathbf{x}_{it}$  are initially assumed to be uncorrelated with  $\varepsilon_{it}$ , an assumption that is relaxed in section 9.4.8. The goal is to consistently estimate  $\gamma_1, \dots, \gamma_p$  and  $\boldsymbol{\beta}$  when  $\alpha_i$  is a fixed effect. The estimators are also consistent if  $\alpha_i$  is a random effect.

The dynamic model (9.3) provides several quite different reasons for correlation in  $y$  over time: 1) directly through  $y$  in preceding periods, called true state dependence; 2) directly through observables  $\mathbf{x}$ , called observed

heterogeneity; and 3) indirectly through the time-invariant individual effect  $\alpha_i$ , called unobserved heterogeneity.

These reasons have substantively different policy implications. For illustration, consider a pure AR(1) time-series model for earnings  $y_{it} = \gamma_1 y_{i,t-1} + \alpha_i + \varepsilon_{it}$  with  $\varepsilon_{it} \simeq 0$  for  $t > 1$ . Suppose in period 1 there is a large positive shock,  $\varepsilon_{i1}$ , leading to a large value for  $y_{i1}$ , moving a low-paid individual to a high-paying job. Then, if  $\gamma_1 \simeq 1$ , earnings will remain high in future years (because  $y_{i,t+1} \simeq y_{it} + \alpha_i$ ). If instead  $\gamma_1 \simeq 0$ , earnings will return to  $\alpha_i$  in future years (because  $y_{i,t+1} \simeq \alpha_i$ ).

Note that the within estimator is inconsistent once lagged regressors are introduced. This is because the within model will have the first regressor  $y_{i,t-1} - \bar{y}_i$  that is correlated with the error  $\varepsilon_{it} - \bar{\varepsilon}_i$  because  $y_{i,t-1}$  is correlated with  $\varepsilon_{i,t-1}$  and hence with  $\bar{\varepsilon}_i$ . Furthermore, IV estimation using lags is not possible, because any lag  $y_{i,s}$  will also be correlated with  $\bar{\varepsilon}_i$  and hence with  $\varepsilon_{it} - \bar{\varepsilon}_i$ . By contrast, although the FD estimator is also inconsistent, IV estimators of the FD model that use appropriate lags of  $y_{it}$  as instruments do lead to consistent parameter estimates.

#### 9.4.2 IV estimation in the FD model

The FD model is

$$\Delta y_{it} = \gamma_1 \Delta y_{i,t-1} + \cdots + \gamma_p \Delta y_{i,t-p} + \Delta \mathbf{x}'_{it} \boldsymbol{\beta} + \Delta \varepsilon_{it}, \quad t = p+1, \dots, T \quad (9.4)$$

We make the crucial assumption that  $\varepsilon_{it}$  are serially uncorrelated, a departure from most analysis to this point that has permitted  $\varepsilon_{it}$  to be correlated over time for a given individual. This assumption is testable, is likely to hold if  $p$  is sufficiently large, and can be relaxed by using `xtdpd`, presented in section [9.4.8](#).

In contrast to a static model, OLS on the first-differenced data produces inconsistent parameter estimates because the regressor  $\Delta y_{i,t-1}$  is correlated with the error  $\Delta \varepsilon_{it}$ , even if  $\varepsilon_{it}$  are serially uncorrelated. For serially uncorrelated  $\varepsilon_{it}$ , the FD model error  $\Delta \varepsilon_{it} = \varepsilon_{it} - \varepsilon_{i,t-1}$  is correlated with  $\Delta y_{i,t-1} = y_{i,t-1} - y_{i,t-2}$  because  $y_{i,t-1}$  depends on  $\varepsilon_{i,t-1}$ . At the same time,  $\Delta \varepsilon_{it}$  is uncorrelated with  $\Delta y_{i,t-k}$  for  $k \geq 2$ , opening up the possibility of IV estimation using lagged variables as instruments.

[Anderson and Hsiao \(1981\)](#) proposed IV estimation using  $y_{i,t-2}$ , which is uncorrelated with  $\Delta\varepsilon_{it}$ , as an instrument for  $\Delta y_{i,t-1}$ . The other lagged dependent variables can be instruments for themselves. The regressors  $x_{it}$  can be used as instruments for themselves if they are strictly exogenous; otherwise, they can also be instrumented as detailed below.

More efficient IV estimators can be obtained by using additional lags of the dependent variable as an instrument; see [Holtz-Eakin, Newey, and Rosen \(1988\)](#). The estimator is then called the Arellano–Bond estimator after [Arellano and Bond \(1991\)](#), who detailed implementation of the estimator and proposed tests of the crucial assumption that  $\varepsilon_{it}$  are serially uncorrelated. Because the instrument set is unbalanced and can be quite complicated, Stata provides the distinct command `xtabond`.

### 9.4.3 The `xtabond` command

The `xtabond` command has the syntax

```
xtabond depvar [indepvars] [if] [in] [, options]
```

The number of lags in the dependent variable,  $p$  in (9.4), is defined by using the `lags (#)` option with the default  $p = 1$ . The regressors are declared in different ways depending on the type of regressor.

First, strictly exogenous regressors are uncorrelated with  $\varepsilon_{it}$ , require no special treatment, are used as instruments for themselves, and are entered as `indepvars`.

Second, predetermined regressors or weakly exogenous regressors are correlated with past errors but are uncorrelated with future errors:  $E(x_{it}\varepsilon_{is}) \neq 0$  for  $s < t$ , and  $E(x_{it}\varepsilon_{is}) = 0$  for  $s \geq t$ . These regressors can be instrumented in the same way that  $y_{i,t-1}$  is instrumented using subsequent lags of  $y_{i,t-1}$ . Specifically,  $x_{it}$  is instrumented by  $x_{i,t-1}, x_{i,t-2}, \dots$ . These regressors are entered by using the `pre(varlist)` option.

Third, a regressor may be contemporaneously endogenous:  $E(x_{it}\varepsilon_{is}) \neq 0$  for  $s \leq t$ , and  $E(x_{it}\varepsilon_{is}) = 0$  for  $s > t$ . Now  $E(x_{it}\varepsilon_{it}) \neq 0$ , so  $x_{i,t-1}$  is no longer a valid instrument in the FD model. The instruments for  $x_{it}$  are now  $x_{i,t-2}, x_{i,t-3}, \dots$ . These regressors are entered by using the `endogenous(varlist)` option.

Finally, additional instruments can be included by using the `inst(varlist)` option.

Potentially, many instruments are available, especially if  $T$  is large. If too many instruments are used, then asymptotic theory provides a poor finite-sample approximation to the distribution of the estimator. The `maxldep(#)` option sets the maximum number of lags of the dependent variable that can be used as instruments. The `maxlags(#)` option sets the maximum number of lags of the predetermined and endogenous variables that can be used as instruments. Alternatively, the `lagstruct(lags, endlags)` suboption can be applied individually to each variable in `pre(varlist)` and `endogenous(varlist)`.

Two different iv estimators can be obtained; see section [7.3](#). The 2SLS estimator, also called the one-step estimator, is the default. Because the model is overidentified, more efficient estimation is possible using optimal generalized method of moments (GMM), also called the two-step estimator because first-step estimation is needed to obtain the optimal weighting matrix used at the second step. The optimal GMM estimator is obtained by using the `twostep` option.

The `vce(robust)` option provides a heteroskedastic-consistent estimate of the variance-covariance matrix of the estimator (VCE). If the  $\varepsilon_{it}$  are serially correlated, the estimator is no longer consistent, so there is no cluster-robust VCE for this case.

Postestimation commands for `xtabond` include `estat abond`, to test the critical assumption of no error correlation, and `estat sargan`, to perform an overidentifying restrictions test; see section [9.4.6](#).

#### 9.4.4 Arellano–Bond estimator: Pure time series

For concreteness, consider an AR(2) model for `lnwage` with no other regressors and seven years of data. Then we have sufficient data to obtain IV estimates in the model

$$\Delta y_{it} = \alpha + \gamma_1 \Delta y_{i,t-1} + \gamma_2 \Delta y_{i,t-2} + \Delta \varepsilon_{it}, \quad t = 4, 5, 6, 7$$

At  $t = 4$ , there are two available instruments,  $y_{i1}$  and  $y_{i2}$ , because these are uncorrelated with  $\Delta \varepsilon_{i4}$ . At  $t = 5$ , there are now three instruments,  $y_{i1}$ ,  $y_{i2}$ , and

$y_{i3}$ , that are uncorrelated with  $\Delta\varepsilon_{i5}$ . Continuing in this manner at  $t = 6$ , there are four instruments,  $y_{i1}, \dots, y_{i4}$ ; and at  $t = 7$ , there are five instruments,  $y_{i1}, \dots, y_{i5}$ . In all, there are  $2 + 3 + 4 + 5 = 14$  available instruments for the two lagged dependent variable regressors. Additionally, the intercept is an instrument for itself. Estimation can be by 2SLS or by the more efficient optimal GMM, which is possible because the model is overidentified. Because the instrument set is unbalanced, it is much easier to use `xtabond` than it is to manually set up the instruments and use `ivregress`.

We apply the estimator to an AR(2) model for the wages data, initially without additional regressors.

```
. * 2SLS or one-step GMM for a pure time-series AR(2) panel model
. qui use mus208psid, clear
. xtabond lwage, lags(2) vce(robust)

Arellano-Bond dynamic panel-data estimation      Number of obs      =      2,380
Group variable: id                            Number of groups   =       595
Time variable: t
                                                Obs per group:
                                                min =           4
                                                avg =           4
                                                max =           4
Number of instruments =      15                  Wald chi2(2)      =     1253.03
                                                Prob > chi2     =     0.0000
One-step results
(Std. err. adjusted for clustering on id)
```

lwage	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
lwage	.5707517	.0333941	17.09	0.000	.5053005	.6362029
L1.	.2675649	.0242641	11.03	0.000	.2200082	.3151216
_cons	1.203588	.164496	7.32	0.000	.8811814	1.525994

Instruments for differenced equation

GMM-type: L(2/.).lwage

Instruments for level equation

Standard: \_cons

There are  $4 \times 595 = 2380$  observations because the first three years of data are lost to construct  $\Delta y_{i,t-2}$ . The results are reported for the original levels model, with the dependent variable  $y_{it}$  and the regressors the lagged dependent variables  $y_{i,t-1}$  and  $y_{i,t-2}$ , even though mechanically the FD model is fit. There are 15 instruments, as already explained, with output `L(2/.)`, meaning that  $y_{i,t-2}, y_{i,t-3}$

, ...,  $y_{i,1}$  are the instruments used for period  $t$ . Wages depend greatly on past wages, with the lag weights summing to  $0.57 + 0.27 = 0.84$ .

The results given are for the 2SLS or one-step estimator. The standard errors reported are robust standard errors that permit the underlying error  $\varepsilon_{it}$  to be heteroskedastic but do not allow for any serial correlation in  $\varepsilon_{it}$ , because then the estimator is inconsistent.

More efficient estimation is possible using optimal or two-step GMM because the model is overidentified. Standard errors reported using the standard textbook formulas for the two-step GMM estimator are downward biased in finite samples. A better estimate of the standard errors, proposed by [Windmeijer \(2005\)](#), can be obtained by using the `vce(robust)` option. As for the one-step estimator, these standard errors permit heteroskedasticity in  $\varepsilon_{it}$ .

Two-step GMM estimation for our data yields

```
. * Optimal or two-step GMM for a pure time-series AR(2) panel model
. xtabond lwage, lags(2) twostep vce(robust)

Arellano-Bond dynamic panel-data estimation      Number of obs      =      2,380
Group variable: id                            Number of groups   =       595
Time variable: t

Number of instruments =      15          Wald chi2(2)      =     1974.40
                                         Prob > chi2      =     0.0000

Two-step results
(Std. err. adjusted for clustering on id)
```

lwage	Coefficient	WC-robust std. err.	z	P> z	[95% conf. interval]
lwage					
L1.	.6095931	.0330542	18.44	0.000	.544808 .6743782
L2.	.2708335	.0279226	9.70	0.000	.2161061 .3255608
_cons	.9182262	.1339978	6.85	0.000	.6555952 1.180857

Instruments for differenced equation

GMM-type: L(2/.).lwage

Instruments for level equation

Standard: \_cons

Here the one-step and two-step estimators have similar estimated coefficients, and the standard errors are also similar, so there is little efficiency gain in two-

step estimation.

For a large  $T$ , the Arellano–Bond method generates many instruments, leading to potential poor performance of asymptotic results. The number of instruments can be restricted by using the `maxldep()` option. For example, we may use only the first available lag, so that just  $y_{i,t-2}$  is the instrument in period  $t$

```
.
. * Reduce the number of instruments for a pure time-series AR(2) panel model
. xtabond lwage, lags(2) vce(robust) maxldep(1)

Arellano-Bond dynamic panel-data estimation      Number of obs      =      2,380
Group variable: id                            Number of groups   =       595
Time variable: t

Obs per group:
    min =           4
    avg =           4
    max =           4

Number of instruments =      5          Wald chi2(2)      =     1372.33
                                         Prob > chi2      =     0.0000

One-step results
                                         (Std. err. adjusted for clustering on id)



| lwage | Coefficient | Robust<br>std. err. | z    | P> z  | [95% conf. interval] |
|-------|-------------|---------------------|------|-------|----------------------|
| lwage |             |                     |      |       |                      |
| L1.   | .4863642    | .1919353            | 2.53 | 0.011 | .110178 .8625505     |
| L2.   | .3647456    | .1661008            | 2.20 | 0.028 | .039194 .6902973     |
| _cons | 1.127609    | .2429357            | 4.64 | 0.000 | .6514633 1.603754    |



Instruments for differenced equation  

GMM-type: L(2/2).lwage  

Instruments for level equation  

Standard: _cons


```

Here there are five instruments:  $y_{i2}$  when  $t = 4$ ,  $y_{i3}$  when  $t = 5$ ,  $y_{i4}$  when  $t = 6$ ,  $y_{i5}$  when  $t = 7$ , and the intercept is an instrument for itself.

In this example, there is considerable loss of efficiency because the standard errors are now about six times larger. This inefficiency disappears if we instead use the `maxldep(2)` option, yielding 8 instruments rather than the original 15.

#### 9.4.5 Arellano–Bond estimator: Additional regressors

We now introduce regressors that are not lagged dependent variables.

We fit a model for `lwage` similar to the model specified in section 9.3. The time-invariant regressors `fem`, `blk`, and `ed` are dropped because they are eliminated after first-differencing. The regressors `occ`, `south`, `smsa`, and `ind` are treated as strictly exogenous. The regressor `wks` appears both contemporaneously and with one lag, and it is treated as predetermined. The regressors `ms` and `union` are treated as endogenous. The first two lags of the dependent variable `lwage` are also regressors.

The model omits one very important regressor, years of work experience (`exp`). For these data, it is difficult to disentangle the separate effects of previous periods' wages and work experience. When both are included, the estimates become very imprecise. Because here we wish to emphasize the role of lagged wages, we exclude work experience from the model.

We fit the model using optimal or two-step GMM and report robust standard errors. The strictly exogenous variables appear as regular regressors. The predetermined and endogenous variables are instead given as options, with restrictions placed on the number of available instruments that are actually used. The dependent variable appears with two lags, and the `maxldep(3)` option is specified so that at most three lags are used as instruments. For example, when  $t = 7$ , the instruments are  $y_{i5}$ ,  $y_{i4}$ , and  $y_{i3}$ . The `pre(wks,lag(1,2))` option is specified so that `wks` and `L1.wks` are regressors, and only two additional lags are to be used as instruments. The `endogenous(ms,lag(0,2))` option is used to indicate that `ms` appears only as a contemporaneous regressor and that at most two additional lags are used as instruments. The `artests(3)` option does not affect the estimation but will affect the postestimation command `estat abond`, as explained in the next section. We have

```

. * Optimal or two-step GMM for a dynamic panel model
. xtabond lwage occ south smsa ind, lags(2) maxldep(3)
>     pre(wks,lag(1,2)) endogenous(ms,lag(0,2))
>     endogenous(union,lag(0,2)) twostep vce(robust) artests(3)

Arellano-Bond dynamic panel-data estimation      Number of obs      =      2,380
Group variable: id                          Number of groups   =       595
Time variable: t

Obs per group:
min =          4
avg =          4
max =          4

Number of instruments =      40          Wald chi2(10)      =    1287.77
                                         Prob > chi2      =    0.0000

Two-step results
(Std. err. adjusted for clustering on id)


```

lwage	Coefficient	WC-robust std. err.	z	P> z	[95% conf. interval]
lwage					
L1.	.611753	.0373491	16.38	0.000	.5385501 .6849559
L2.	.2409058	.0319939	7.53	0.000	.1781989 .3036127
wks					
--.	-.0159751	.0082523	-1.94	0.053	-.0321493 .000199
L1.	.0039944	.0027425	1.46	0.145	-.0013807 .0093695
ms	.1859324	.144458	1.29	0.198	-.0972 .4690649
union	-.1531329	.1677842	-0.91	0.361	-.4819839 .1757181
occ	-.0357509	.0347705	-1.03	0.304	-.1038999 .032398
south	-.0250368	.2150806	-0.12	0.907	-.446587 .3965134
smsa	-.0848223	.0525243	-1.61	0.106	-.187768 .0181235
ind	.0227008	.0424207	0.54	0.593	-.0604422 .1058437
_cons	1.639999	.4981019	3.29	0.001	.6637377 2.616261

#### Instruments for differenced equation

GMM-type: L(2/4).lwage L(1/2).L.wks L(2/3).ms L(2/3).union  
 Standard: D.occ D.south D.smsa D.ind

#### Instruments for level equation

Standard: \_cons

With the inclusion of additional regressors, the coefficients of the lagged dependent variables have changed little, and the standard errors are about 10–15% higher. The additional regressors are all statistically insignificant at 5%. By contrast, some are statistically significant using the within estimator for a static model that does not include the lagged dependent variables.

The output explains the instruments used. For example, L(2/4).lwage means that  $lwage_{i,t-2}$ ,  $lwage_{i,t-3}$ , and  $lwage_{i,t-4}$  are used as instruments, provided they are available. In the initial period  $t = 4$ , only the first two of these are available,

whereas in  $t = 5, 6, 7$ , all three are available for a total of  $2 + 3 + 3 + 3 = 11$  instruments. By similar analysis, `L(1/2).L.wks`, `L(2/3).ms`, and `L(2/3).union` each provide 8 instruments, and there are 5 standard instruments. In all, there are  $11 + 8 + 8 + 8 + 5 = 40$  instruments, as stated at the top of the output.

#### 9.4.6 Specification tests

For consistent estimation, the `xtabond` estimators require that the error  $\varepsilon_{it}$  be serially uncorrelated. This assumption is testable.

Specifically, if  $\varepsilon_{it}$  are serially uncorrelated, then  $\Delta\varepsilon_{it}$  are correlated with  $\Delta\varepsilon_{i,t-1}$  because  $\text{Cov}(\Delta\varepsilon_{it}, \Delta\varepsilon_{i,t-1}) = \text{Cov}(\varepsilon_{it} - \varepsilon_{i,t-1}, \varepsilon_{i,t-1} - \varepsilon_{i,t-2}) = -\text{Cov}(\varepsilon_{i,t-1}, \varepsilon_{i,t-1}) \neq 0$ . But  $\Delta\varepsilon_{it}$  will not be correlated with  $\Delta\varepsilon_{i,t-k}$  for  $k \geq 2$ . A test of whether  $\Delta\varepsilon_{it}$  are correlated with  $\Delta\varepsilon_{i,t-k}$  for  $k \geq 2$  can be calculated based on the correlation of the fitted residuals  $\widehat{\Delta\varepsilon}_{it}$ . This is performed by using the `estat abond` command.

The default is to test to lag 2, but here we also test the third lag. This can be done in two ways. One way is to use `estat abond` with the `artests(3)` option, which leads to recalculation of the estimator defined in the preceding `xtabond` command. Alternatively, we can include the `artests(3)` option in `xtabond`, in which case we simply use `estat abond` and no recalculation is necessary.

In our case, the `artests(3)` option was included in the preceding `xtabond` command. We obtain

```
. * Test whether error is serially correlated
. estat abond
Arellano-Bond test for zero autocorrelation in first-differenced errors
H0: No autocorrelation

Order      z    Prob > z

```

Order	z	Prob > z
1	-4.5244	0.0000
2	-1.6041	0.1087
3	.35729	0.7209

The null hypothesis that  $\text{Cov}(\Delta\varepsilon_{it}, \Delta\varepsilon_{i,t-k}) = 0$  for  $k = 1, 2, 3$  is rejected at a level of 0.05 if  $p < 0.05$ . As explained above, if  $\varepsilon_{it}$  are serially uncorrelated, we expect to reject at order 1 but not at higher orders. This is indeed the case. We reject at order 1 because  $p = 0.000$ . At order 2,  $\Delta\varepsilon_{it}$  and  $\Delta\varepsilon_{i,t-2}$  are serially uncorrelated because  $p = 0.109 > 0.05$ . Similarly, at order 3, there is no

evidence of serial correlation because  $p = 0.721 > 0.05$ . There is no serial correlation in the original error  $\varepsilon_{it}$ , as desired.

A second specification test is a test of overidentifying restrictions; see section [7.4.8](#). Here 40 instruments were used to estimate 11 parameters, so there are 29 overidentifying restrictions. The `estat sargan` command implements the test. This command is not implemented after `xtabond` if the `vce(robust)` option is used, because the test is then invalid because it requires that the errors  $\varepsilon_{it}$  be independent and identically distributed (i.i.d.). We therefore need to first run `xtabond` without this option. We have

```
. * Test of overidentifying restrictions (first estimate with no vce(robust))
. qui xtabond lwage occ south smsa ind, lags(2) maxldep(3)
>     pre(wks,lag(1,2)) endogenous(ms,lag(0,2))
>     endogenous(union,lag(0,2)) twostep artests(3)

. estat sargan
Sargan test of overidentifying restrictions
H0: Overidentifying restrictions are valid
      chi2(29)      =  39.87571
      Prob > chi2  =    0.0860
```

The null hypothesis that the population moment conditions are correct is not rejected, because  $p = 0.086 > 0.05$ .

#### 9.4.7 The `xtdpdsys` command

The Arellano–Bond estimator uses an IV estimator based on the assumption that  $E(y_{is}\Delta\varepsilon_{it}) = 0$  for  $s \leq t - 2$  in [\(9.3\)](#), so that the lags  $y_{i,t-2}, y_{i,t-3}, \dots$  can be used as instruments in the first-differenced [\(9.4\)](#). Several articles suggest using additional moment conditions to obtain an estimator with improved precision and better finite-sample properties. In particular, [Arellano and Bover \(1995\)](#) and [Blundell and Bond \(1998\)](#) consider using the additional condition  $E(\Delta y_{i,t-1}\varepsilon_{it}) = 0$  so that we also incorporate the levels [\(9.3\)](#) and use as an instrument  $\Delta y_{i,t-1}$ . Similar additional moment conditions can be added for endogenous and predetermined variables, whose FDS can be used as instruments.

This estimator, often called the systems estimator, is performed by using the command `xtdpdsys`. It is also performed by using the community-contributed `xtabond2` command. The syntax is exactly the same as that for `xtabond`.

We refit the model of section [9.4.5](#) using `xtdpdsys` rather than `xtabond`.

```

. * Arellano/Bover or Blundell/Bond for a dynamic panel model
. xtdpdsys lwage occ south smsa ind, lags(2) maxldep(3)
>     pre(wks,lag(1,2)) endogenous(ms,lag(0,2))
>     endogenous(union,lag(0,2)) twostep vce(robust) artests(3)

System dynamic panel-data estimation
Number of obs      =      2,975
Group variable: id          Number of groups =       595
Time variable: t

Obs per group:
min =           5
avg =           5
max =           5

Number of instruments =      60
Wald chi2(10)      =    2270.88
Prob > chi2        =     0.0000

```

#### Two-step results

lwage	Coefficient	WC-robust std. err.	z	P> z	[95% conf. interval]
lwage					
L1.	.6017533	.0291502	20.64	0.000	.5446199 .6588866
L2.	.2880537	.0285319	10.10	0.000	.2321322 .3439752
wks					
--.	-.0014979	.0056143	-0.27	0.790	-.0125017 .009506
L1.	.0006786	.0015694	0.43	0.665	-.0023973 .0037545
ms	.0395337	.0558543	0.71	0.479	-.0699386 .1490061
union	-.0422409	.0719919	-0.59	0.557	-.1833423 .0988606
occ	-.0508803	.0331149	-1.54	0.124	-.1157843 .0140237
south	-.1062817	.083753	-1.27	0.204	-.2704346 .0578713
smsa	-.0483567	.0479016	-1.01	0.313	-.1422422 .0455288
ind	.0144749	.031448	0.46	0.645	-.0471621 .0761118
_cons	.9584113	.3632287	2.64	0.008	.2464961 1.670327

#### Instruments for differenced equation

GMM-type: L(2/4).lwage L(1/2).L.wks L(2/3).ms L(2/3).union  
 Standard: D.occ D.south D.smsa D.ind

#### Instruments for level equation

GMM-type: LD.lwage LD.wks LD.ms LD.union  
 Standard: \_cons

There are now 60 instruments rather than 40 instruments because the lagged FDS in `lwage`, `wks`, `ms`, and `union` are available for each of the 5 periods  $t = 3, \dots, 7$ . There is some change in estimated coefficients. More noticeable is a reduction in standard errors of 10–60%, reflecting greater precision because of the additional moment conditions.

The procedure assumes the errors  $\varepsilon_{it}$  are serially uncorrelated. This assumption can be tested by using the `estat abond` postestimation command, and from output not given, this test confirms that the errors are serially uncorrelated

here. If the `xtdpdsys` command is run with the default standard errors, the `estat sargan` command can be used to test the overidentifying conditions.

#### 9.4.8 The `xtdpd` command

The preceding estimators and commands require that the model errors  $\varepsilon_{it}$  be serially uncorrelated. If this assumption is rejected (it is testable by using the `estat abond` command), then one possibility is to add more lags of the dependent variable as regressors in the hope that this will eliminate any serial correlation in the error.

An alternative is to use the `xtdpd` command that allows  $\varepsilon_{it}$  to follow a moving-average (MA) process of low order. This command also allows predetermined variables to have a more complicated structure.

For `xtdpd`, a very different syntax is used to enter all the variables and instruments in the model; see [XT] **xtdpd**. Essentially, one specifies a variable list with all model regressors (lagged dependent, exogenous, predetermined, and endogenous), followed by options that specify instruments. For exogenous regressors, the `div()` option is used, and for other types of regressors, the `dgmmiv()` option is used with the explicit statement of the lags of each regressor to be used as instruments. Instruments for the levels equation, used in the `xtdpdsys` command, can also be specified with the `lgmmiv()` option.

As an example, we provide without explanation an `xtdpd` command that exactly reproduces the `xtdpdsys` command of the previous section. We have

```

. * Use of xtdpd to exactly reproduce the previous xtdpdsys command
. xtdpd L(0/2).lwage L(0/1).wks occ south smsa ind ms union,
>     div(occ south smsa ind) dgmmiv(lwage, lagrange(2 4))
>     dgmmiv(ms union, lagrange(2 3)) dgmmiv(L.wks, lagrange(1 2))
>     lgmmiv(lwage wks ms union) twostep vce(robust) artests(3)

Dynamic panel-data estimation
Number of obs      =      2,975
Group variable: id
Number of groups   =       595
Time variable: t
Obs per group:
min =           5
avg =           5
max =           5
Number of instruments =      60
Wald chi2(10)      =    2270.88
Prob > chi2        =     0.0000

Two-step results
(Std. err. adjusted for clustering on id)


```

lwage	Coefficient	WC-robust std. err.	z	P> z	[95% conf. interval]
lwage					
L1.	.6017533	.0291502	20.64	0.000	.5446199 .6588866
L2.	.2880537	.0285319	10.10	0.000	.2321322 .3439752
wks					
--.	-.0014979	.0056143	-0.27	0.790	-.0125017 .009506
L1.	.0006786	.0015694	0.43	0.665	-.0023973 .0037545
occ	-.0508803	.0331149	-1.54	0.124	-.1157843 .0140237
south	-.1062817	.083753	-1.27	0.204	-.2704346 .0578713
smsa	-.0483567	.0479016	-1.01	0.313	-.1422422 .0455288
ind	.0144749	.031448	0.46	0.645	-.0471621 .0761118
ms	.0395337	.0558543	0.71	0.479	-.0699386 .1490061
union	-.0422409	.0719919	-0.59	0.557	-.1833423 .0988606
_cons	.9584113	.3632287	2.64	0.008	.2464961 1.670327

#### Instruments for differenced equation

GMM-type: L(2/4).lwage L(2/3).ms L(2/3).union L(1/2).L.wks

Standard: D.occ D.south D.smsa D.ind

#### Instruments for level equation

GMM-type: LD.lwage LD.wks LD.ms LD.union

Standard: \_cons

Now suppose that the error  $\varepsilon_{it}$  in (9.3) is MA(1), so that  $\varepsilon_{it} = \eta_{it} + \delta\eta_{i,t-1}$ , where  $\eta_{it}$  is i.i.d. Then  $y_{i,t-2}$  is no longer a valid instrument, but  $y_{i,t-3}$  and further lags are. Also, for the level equation,  $\Delta y_{i,t-1}$  is no longer a valid instrument, but  $\Delta y_{i,t-2}$  is valid. We need to change the `dgmmiv()` and `lgmmiv()` options for `lwage`. The command becomes

```

. * Previous command if model error is MA(1)
. xtdpd L(0/2).lwage L(0/1).wks occ south smsa ind ms union,
> div(occ south smsa ind) dgmmiv(lwage, lagrange(3 4))
> dgmmiv(ms union, lagrange(2 3)) dgmmiv(L.wks, lagrange(1 2))
> lgmmiv(L.lwage wks ms union) twostep vce(robust) artests(3)
    (output omitted)

```

### 9.4.9 The xtabond2 command

For fitting FE linear dynamic panel models, possibly with serially correlated errors, using short panel data, the `xtabond2` command ([Roodman 2009](#)) provides an alternative to the `xtabond` command. Additionally, [Roodman \(2009\)](#) provides a very detailed discussion of the Arellano–Bond method that is very useful to read even if one uses the `xtabond` command.

The `xtabond2` command and the computer output it generates have several attractive features. To eliminate the fixed effects, the command as an option uses the orthogonal deviations transformation or Helmert transformation, which is a more data-saving method for eliminating the fixed effects than the FD transformation if data are not available in all time periods. To limit the number of instruments it permits both “GMM-style” and “IV-style” specification of the lags and options for limiting the number of moment conditions used. And the `xtabond2` command provides an option to obtain Windmeijer-corrected cluster-robust efficient two-step standard errors ([Windmeijer 2005](#)).

We illustrate the syntax and use of `xtabond2` by reestimating some of the specifications considered in previous sections, beginning with two-step GMM estimation of the pure AR(2) model with intercept estimated in section [9.4.4](#) and with instruments the second and later lags of the dependent variable.

Here we use the option `h(1)`, which obtains 2SLS estimates at the first stage, and the `small` option, which bases inference on the  $t$  and  $F$  distributions rather than the normal and chi-squared distributions.

```

. * xtabond2: Two-step GMM for a pure time-series AR(2) panel model
. xtabond2 lwage L.lwage L2.lwage, gmmstyle(L2.lwage) h(1) small twostep robust
Favoring space over speed. To switch, type or click on mata: mata set matafavor
> speed, perm.
Warning: Two-step estimated covariance matrix of moments is singular.
Using a generalized inverse to calculate optimal weighting matrix for two-step
> estimation.
Difference-in-Sargan/Hansen statistics may be negative.

```

Dynamic panel-data estimation, two-step system GMM

Group variable: id	Number of obs	=	2975
Time variable : t	Number of groups	=	595
Number of instruments = 15	Obs per group: min	=	5
F(2, 594) = 2.00e+06	avg	=	5.00
Prob > F = 0.000	max	=	5

lwage	Coefficient	Corrected				[95% conf. interval]
		std. err.	t	P> t		
lwage	.8034441	.0763937	10.52	0.000	.6534096	.9534787
L1.	.0938402	.0672123	1.40	0.163	-.0381624	.2258429
_cons	.7877662	.1191901	6.61	0.000	.553681	1.021852

Instruments for first differences equation

GMM-type (missing=0, separate instruments for each period unless collapsed)  
L(1/6).L2.lwage

Instruments for levels equation

Standard

\_cons

GMM-type (missing=0, separate instruments for each period unless collapsed)  
D.L2.lwage

---

Arellano-Bond test for AR(1) in first differences: z = -3.06 Pr > z = 0.002  
Arellano-Bond test for AR(2) in first differences: z = 0.59 Pr > z = 0.558

---

Sargan test of overid. restrictions: chi2(12) = 23.35 Prob > chi2 = 0.025  
(Not robust, but not weakened by many instruments.)

Hansen test of overid. restrictions: chi2(12) = 16.12 Prob > chi2 = 0.186  
(Robust, but weakened by many instruments.)

Difference-in-Hansen tests of exogeneity of instrument subsets:

GMM instruments for levels

Hansen test excluding group: chi2(8) = 12.52 Prob > chi2 = 0.129  
Difference (null H = exogenous): chi2(4) = 3.60 Prob > chi2 = 0.462

The same number of instruments is used as in section [9.4.4](#). The number of time periods used is five rather than four because the default for the `xtabond2` command is to use the systems estimator of section [9.4.7](#), which adds a levels equation. This also leads to estimates that differ from those in section [9.4.4](#). The tests of overidentifying restrictions are included with the warning that they are less reliable in a setting with many instruments.

Our second reported specification adds exogenous variables (`occ south smsa ind`), endogenous variables (`wks L.wks ms union`), and four FE time dummies (`tdum3–tdum6`) because estimation uses five years of data and one of the time dummies is dropped to avoid the dummy variables trap. Instruments are generated using the GMM style for lagged endogenous variables as instruments, as well as the IV style applied to time dummies. Here the GMM-style option means that different lagged instruments apply to different periods, and the IV-style option means that the exogenous variable is a common instrument for all periods. We obtain

```
. * xtabond2: Two-step GMM for dynamic model with exogenous & endogenous vars.
. xtabond2 lwage L.lwage L2.lwage occ south smsa ind wks L.wks ms union
>      tdum3-tdum6, gmmstyle(L.(L.lwage wks ms union))
>      ivstyle(t, equation(level)) twostep small
Favoring space over speed. To switch, type or click on mata: mata set matafavor
> speed, perm.
Warning: Two-step estimated covariance matrix of moments is singular.
Using a generalized inverse to calculate optimal weighting matrix for two-step
> estimation.
Difference-in-Sargan/Hansen statistics may be negative.
```

Dynamic panel-data estimation, two-step system GMM

Group variable: id		Number of obs	=	2975
Time variable : t		Number of groups	=	595
Number of instruments = 73		Obs per group: min	=	5
F(14, 594) = 27435.06		avg	=	5.00
Prob > F = 0.000		max	=	5

lwage	Coefficient	Std. err.	t	P> t	[95% conf. interval]
lwage					
L1.	.623573	.0659086	9.46	0.000	.4941307 .7530153
L2.	.026329	.0461591	0.57	0.569	-.0643258 .1169838
occ	-.3792783	.085034	-4.46	0.000	-.5462821 -.2122744
south	-.3445325	.1500747	-2.30	0.022	-.639274 -.049791
smsa	-.25237	.1264818	-2.00	0.046	-.5007758 -.0039642
ind	.2997457	.1158891	2.59	0.010	.0721435 .527348
wks					
---	-.0012231	.0041107	-0.30	0.766	-.0092964 .0068503
L1.	.0009459	.0012505	0.76	0.450	-.0015101 .0034019
ms	.0579029	.0518024	1.12	0.264	-.0438353 .1596411
union	-.1356566	.0666732	-2.03	0.042	-.2666004 -.0047128
tdum3	-.0963978	.0229935	-4.19	0.000	-.1415563 -.0512393
tdum4	-.0717887	.017493	-4.10	0.000	-.1061443 -.0374332
tdum5	-.0497818	.0139664	-3.56	0.000	-.0772113 -.0223522
tdum6	-.0239232	.0101656	-2.35	0.019	-.0438882 -.0039583
_cons	2.836121	.4131195	6.87	0.000	2.024768 3.647473

```

Warning: Uncorrected two-step standard errors are unreliable.

Instruments for first differences equation
  GMM-type (missing=0, separate instruments for each period unless collapsed)
    L(1/6).(L2.lwage L.wks L.ms L.union)

Instruments for levels equation
  Standard
    t
    _cons
  GMM-type (missing=0, separate instruments for each period unless collapsed)
    D.(L2.lwage L.wks L.ms L.union)

```

---

```

Arellano-Bond test for AR(1) in first differences: z = -4.07 Pr > z = 0.000
Arellano-Bond test for AR(2) in first differences: z = 0.94 Pr > z = 0.349

```

---

```

Sargan test of overid. restrictions: chi2(58) = 66.44 Prob > chi2 = 0.209
(Not robust, but not weakened by many instruments.)
Hansen test of overid. restrictions: chi2(58) = 52.97 Prob > chi2 = 0.662
(Robust, but weakened by many instruments.)

```

Difference-in-Hansen tests of exogeneity of instrument subsets:

```

GMM instruments for levels
  Hansen test excluding group: chi2(39) = 30.70 Prob > chi2 = 0.826
  Difference (null H = exogenous): chi2(19) = 22.26 Prob > chi2 = 0.271
  iv(t, eq(level))
  Hansen test excluding group: chi2(57) = 46.66 Prob > chi2 = 0.834
  Difference (null H = exogenous): chi2(1) = 6.31 Prob > chi2 = 0.012

```

The number of instruments has now jumped to 73, which makes the overidentification tests even less reliable. The endogenous and lagged endogenous variables have little explanatory power in this case.

Our third example repeats the previous calculation but uses one of the options to limit the number of lags used to form instruments to 2 and 3.

```

. * xtabond2: Two-step GMM for dynamic model with limit on the lags generating IV
. xtabond2 lwage L.lwage L2.lwage occ south smsa ind wks L.wks ms union
> tdum3-tdum6, gmmstyle(L.(L.lwage wks ms union), laglimits(2 3)) twostep small
Favoring space over speed. To switch, type or click on mata: mata set matafavor
> speed, perm.
Warning: Two-step estimated covariance matrix of moments is singular.
Using a generalized inverse to calculate optimal weighting matrix for two-step
> estimation.
Difference-in-Sargan/Hansen statistics may be negative.

```

Dynamic panel-data estimation, two-step system GMM

Group variable: id	Number of obs	=	2975
Time variable : t	Number of groups	=	595
Number of instruments = 42	Obs per group: min	=	5
F(14, 594) = 19203.93	avg	=	5.00
Prob > F = 0.000	max	=	5

lwage	Coefficient	Std. err.	t	P> t	[95% conf. interval]
lwage					
L1.	.4911215	.122592	4.01	0.000	.2503549 .731888
L2.	.0673887	.1155809	0.58	0.560	-.1596082 .2943856
occ	-.4276428	.1332561	-3.21	0.001	-.6893532 -.1659324
south	-.4102166	.2282793	-1.80	0.073	-.8585493 .0381162
smsa	-.2879704	.219564	-1.31	0.190	-.7191866 .1432458
ind	.1600471	.2229759	0.72	0.473	-.2778698 .5979641
wks					
--.	-.000818	.0100056	-0.08	0.935	-.0204687 .0188327
L1.	.0106464	.005731	1.86	0.064	-.000609 .0219019
ms	.0832632	.110451	0.75	0.451	-.1336587 .3001851
union	-.1824594	.1151066	-1.59	0.113	-.4085248 .043606
tdum3	-.1302655	.0426063	-3.06	0.002	-.2139427 -.0465882
tdum4	-.1014275	.032352	-3.14	0.002	-.1649657 -.0378893
tdum5	-.0712044	.0241933	-2.94	0.003	-.1187193 -.0236896
tdum6	-.0377894	.0135913	-2.78	0.006	-.0644822 -.0110966
_cons	3.115346	.8056482	3.87	0.000	1.53308 4.697611

```

Warning: Uncorrected two-step standard errors are unreliable.

Instruments for first differences equation
    GMM-type (missing=0, separate instruments for each period unless collapsed)
        L(2/3).(L2.lwage L.wks L.ms L.union)

Instruments for levels equation
    Standard
        _cons
    GMM-type (missing=0, separate instruments for each period unless collapsed)
        DL.(L2.lwage L.wks L.ms L.union)

Arellano-Bond test for AR(1) in first differences: z = -2.98 Pr > z = 0.003
Arellano-Bond test for AR(2) in first differences: z = 0.21 Pr > z = 0.833

Sargan test of overid. restrictions: chi2(27) = 36.10 Prob > chi2 = 0.113
    (Not robust, but not weakened by many instruments.)
Hansen test of overid. restrictions: chi2(27) = 26.41 Prob > chi2 = 0.496
    (Robust, but weakened by many instruments.)

Difference-in-Hansen tests of exogeneity of instrument subsets:
    GMM instruments for levels
        Hansen test excluding group: chi2(12) = 6.53 Prob > chi2 = 0.887
        Difference (null H = exogenous): chi2(15) = 19.88 Prob > chi2 = 0.177

```

As a result, the number of instruments now falls to 42. The standard errors are now larger in several cases, so there is some loss of efficiency.

#### 9.4.10 Dynamic systems with fixed effects

[Arellano and Bond \(1991\)](#) considered GMM estimation of a single dynamic equation with individual specific fixed effects. The earlier article by [Holtz-Eakin, Newey, and Rosen \(1988\)](#) actually considered GMM estimation of systems of dynamic equations, or vector autoregressive models, with individual specific fixed effects in each equation.

The community-contributed `pvar` command ([Abrigo and Love 2016](#)) implements the systems estimator of [Holtz-Eakin, Newey, and Rosen \(1988\)](#). In addition to providing model estimates, the `pvar` command produces impulse-response functions.

## 9.5 Long panels

The methods to this point have focused on short panels. Now we consider long panels with many time periods.

Sections [9.5.1–9.5.5](#) focus on the case of relatively few individuals ( $N$  is small and  $T \rightarrow \infty$ ). Examples are data on a few regions, firms, or industries followed for many time periods. Then individual fixed effects, if desired, can be easily handled by including dummy variables for each individual as regressors. Instead, the focus is on more efficient GLS estimation under richer models of the error process than those specified in the short-panel case.

Some subsequent subsections also consider the case where both  $N \rightarrow \infty$  and  $T \rightarrow \infty$ . The section ends with brief coverage of panel data with unit roots and cointegration.

### 9.5.1 Long-panel dataset

The dataset used is a U.S. state–year panel from [Baltagi, Griffin, and Xiong \(2000\)](#) on annual cigarette consumption and price for U.S. states over 30 years. The ultimate goal is to measure the responsiveness of per capita cigarette consumption to real cigarette prices. Price varies across states, due in large part to different levels of taxation, as well as over time.

The original data were for  $N = 46$  states and  $T = 30$ , and it is not clear whether we should treat  $N \rightarrow \infty$ , as we have done to date, or  $T \rightarrow \infty$ , or both. This situation is not unusual for a panel that uses aggregated regional data over time. To make explicit that we are considering  $T \rightarrow \infty$ , we use data from only  $N = 10$  states, similar to many countries where there may be about 10 major regions (states or provinces).

`mus209cigar.dta` has the following data:

```

. * Description of cigarette dataset
. qui use mus209cigar, clear
. describe

Contains data from mus209cigar.dta
Observations:           300
Variables:              6
A.C.Cameron & P.K.Trivedi
(2022): Microeconometrics Using
Stata, 2e
1 Sep 2020 16:38

```

Variable name	Storage type	Display format	Value label	Variable label
state	float	%9.0g		U.S. state
year	float	%9.0g		Year 1963 to 1992
lnp	float	%9.0g		Log state real price of pack of cigarettes
lnpmin	float	%9.0g		Log of min real price in adjoining states
lnc	float	%9.0g		Log state cigarette sales in packs per capita
lny	float	%9.0g		Log state per capita disposable income

Sorted by:

There are 300 observations, so each state–year pair is a separate observation because  $10 \times 30 = 300$ . The quantity demanded (`lnc`) will depend on price (`lnp`), price of a substitute (`lnpmin`), and income (`lny`).

Descriptive statistics can be obtained by using `summarize`:

```

. * Summary of cigarette dataset
. summarize, separator(6)

```

Variable	Obs	Mean	Std. dev.	Min	Max
state	300	5.5	2.87708	1	10
year	300	77.5	8.669903	63	92
lnp	300	4.518424	.1406979	4.176332	4.96916
lnpmin	300	4.4308	.1379243	4.0428	4.831303
lnc	300	4.792591	.2071792	4.212128	5.690022
lny	300	8.731014	.6942426	7.300023	10.0385

The variables `state` and `year` have the expected ranges. The variability in per capita cigarette sales (`lnc`) is actually greater than the variability in price (`lnp`), with respective standard deviations of 0.21 and 0.14. All variables are observed for all 300 observations, so the panel is indeed balanced.

## 9.5.2 Pooled OLS and pooled feasible GLS

A natural starting point is the two-way-effects model

$y_{it} = \alpha_i + \gamma_t + \mathbf{x}'_{it}\beta + \varepsilon_{it}$ . When the panel has few individuals relative to the number of periods, the individual effects  $\alpha_i$  (here state effects) can be incorporated into  $\mathbf{x}_{it}$  as dummy-variable regressors. Then there are too many time effects  $\gamma_t$  (here year effects). Rather than trying to control for these in ways analogous to the use of `xtreg` in the short-panel case, we can usually take advantage of the natural ordering of time (as opposed to individuals) and simply include a linear or quadratic trend in time.

We therefore focus on the pooled model

$$y_{it} = \mathbf{x}'_{it}\beta + u_{it}, \quad i = 1, \dots, N, \quad t = 1, \dots, T$$

where the regressors  $\mathbf{x}_{it}$  include an intercept, often time and possibly time squared, and possibly a set of individual indicator variables. We assume that  $N$  is quite small relative to  $T$ .

We consider pooled OLS and pooled feasible generalized least-squares (PFGLS) of this model under a variety of assumptions about the error  $u_{it}$ . In the short-panel case, it was possible to obtain standard errors that control for serial correlation in the error without explicitly stating a model for serial correlation. Instead, we could use cluster-robust standard errors, given a small  $T$  and  $N \rightarrow \infty$ . Now, however,  $T$  is large relative to  $N$ , and it is necessary to specify a model for serial correlation in the error. Also, given that  $N$  is small, it is possible to relax the assumption that  $u_{it}$  is independent over  $i$ .

## 9.5.3 The `xtpcse` and `xtgls` commands

The `xtpcse` and `xtgls` commands are more suited than `xtgee` for pooled OLS and GLS when data are from a long panel. They allow the error  $u_{it}$  in the model to be correlated over  $i$ , allow the use of an AR(1) model for  $u_{it}$  over  $t$ , and allow  $u_{it}$  to be heteroskedastic. At the greatest level of generality,

$$u_{it} = \rho_i u_{i,t-1} + \varepsilon_{it} \quad (9.5)$$

where  $\varepsilon_{it}$  are serially uncorrelated but are correlated over  $i$  with  $\text{Cor}(\varepsilon_{it}, \varepsilon_{is}) = \sigma_{ts}$ .

The `xtpcse` command yields (long) panel-corrected standard errors for the pooled OLS estimator, as well as for a pooled least-squares estimator with an AR(1) model for  $u_{it}$ . The syntax is

```
xtpcse depvar [indepvars] [if] [in] [weight] [, options]
```

The `correlation()` option determines the type of pooled estimator. Pooled OLS is obtained by using `correlation(independent)`. The pooled AR(1) estimator with general  $\rho_i$  is obtained by using `correlation(psar1)`. With a balanced panel,  $y_{it} - \hat{\rho}_i y_{it,t-1}$  is regressed on  $\mathbf{x}_{it}^* = \mathbf{x}_{it} - \hat{\rho} \mathbf{x}_{it,t-1}$  for  $t > 1$ , whereas  $\sqrt{(1 - \hat{\rho}_i)^2} y_{i1}$  is regressed on  $\sqrt{(1 - \hat{\rho}_i)^2} \mathbf{x}_{i1}$  for  $t = 1$ . The pooled estimator with AR(1) error and  $\rho_i = \rho$  is obtained by using `correlation(ar1)`. Then  $\hat{\rho}$ , calculated as the average of the  $\hat{\rho}_i$ , is used.

In all cases, panel-corrected standard errors that allow heteroskedasticity and correlation over  $i$  are reported, unless the `hetonly` option is used, in which case independence over  $i$  is assumed, or the `independent` option is used, in which case  $\varepsilon_{it}$  is i.i.d.

The `xtgls` command goes further and obtains PFGLS estimates and associated standard errors assuming the model for the errors is the correct model. The estimators are more efficient asymptotically than those from `xtpcse`, if the model is correctly specified. The command has the usual syntax:

```
xtgls depvar [indepvars] [if] [in] [weight] [, options]
```

The `panels()` option specifies the error correlation across individuals, where for our data an individual is a state. The `panels(iid)` option specifies  $u_{it}$  to be i.i.d., in which case the pooled OLS estimator is obtained. The `panels(heteroskedastic)` option specifies  $u_{it}$  to be independent with a variance of  $E(u_{it}^2) = \sigma_i^2$  that can be different for each individual. Because there are many observations for each individual,  $\sigma_i^2$  can be consistently

estimated. The `panels(correlated)` option additionally allows correlation across individuals, with independence over time for a given individual, so that  $E(u_{it}u_{jt}) = \sigma_{ij}$ . This option requires that  $T > N$ .

The `corr()` option specifies the serial correlation of errors for each individual state. The `corr(independent)` option specifies  $u_{it}$  to be serially uncorrelated. The `corr(ar1)` option permits AR(1) autocorrelation of the error with  $u_{it} = \rho u_{i,t-1} + \varepsilon_{it}$ , where  $\varepsilon_{it}$  is i.i.d. The `corr(psar1)` option relaxes the assumption of a common AR(1) parameter to allow  $u_{it} = \rho_i u_{i,t-1} + \varepsilon_{it}$ . The `rhotype()` option provides various methods to compute this AR(1) parameter. The default estimator is two-step feasible GLS, whereas the `igls` option uses iterated feasible GLS. The `force` option enables estimation even if observations are unequally spaced over time.

Additionally, we illustrate the community-contributed `xtscc` command ([Hoechle 2007](#)). This generalizes `xtpcse` by applying the method of [Driscoll and Kraay \(1998\)](#) to obtain Newey–West-type standard errors for pooled OLS that allow autocorrelated errors of general form, rather than restricting errors to be AR(1). Error correlation across panels, often called spatial correlation, is assumed. The error is allowed to be serially correlated for  $m$  lags. The default is for the program to determine  $m$ . Alternatively,  $m$  can be specified using the `lags(m)` option.

#### 9.5.4 Application of the `xtgls`, `xtpcse`, and `xtscc` commands

As an example, we begin with a PFGLS estimator that uses the most flexible model for the error  $u_{it}$ , with flexible correlation across states and a distinct AR(1) process for the error in each state. In principle, this is the best estimator to use, but in practice, when  $T$  is not much larger than  $N$ , there can be finite-sample bias in the estimators and standard errors; see [Beck and Katz \(1995\)](#). Then it is best, at the least, to use the more restrictive `corr(ar1)` rather than `corr(psar1)`.

We obtain

```

. * Pooled GLS with error correlated across states and state-specific AR(1)
. xtset state year
Panel variable: state (strongly balanced)
Time variable: year, 63 to 92
    Delta: 1 unit
. xtgls lnc lnp lny lnpmin year, panels(correlated) corr(psar1)
Cross-sectional time-series FGLS regression
Coefficients: generalized least squares
Panels: heteroskedastic with cross-sectional correlation
Correlation: panel-specific AR(1)
Estimated covariances      =      55          Number of obs      =      300
Estimated autocorrelations =      10          Number of groups   =       10
Estimated coefficients     =       5          Time periods     =       30
                                         Wald chi2(4)      =     342.15
                                         Prob > chi2      =     0.0000

```

	lnc	Coefficient	Std. err.	z	P> z	[95% conf. interval]
	lnp	-.3260683	.0218214	-14.94	0.000	-.3688375 -.2832991
	lny	.4646236	.0645149	7.20	0.000	.3381768 .5910704
	lnpmin	.0174759	.0274963	0.64	0.525	-.0364159 .0713677
	year	-.0397666	.0052431	-7.58	0.000	-.0500429 -.0294902
	_cons	5.157994	.2753002	18.74	0.000	4.618416 5.697573

All regressors have the expected effects. The estimated price elasticity of demand for cigarettes is  $-0.326$ ; the income elasticity is  $0.465$ ; demand declines by 4% per year (the coefficient of `year` is a semielasticity because the dependent variable is in logs); and a higher minimum price in adjoining states increases demand in the current state. There are 10 states, so there are  $10 \times 11/2 = 55$  unique entries in the  $10 \times 10$  contemporaneous error covariance matrix, and 10 autocorrelation parameters  $\rho_i$  are estimated.

We now use `xtpcse`, `xtgls`, and community-contributed `xtscc` to obtain the following pooled estimators and associated standard errors: 1) pooled OLS with i.i.d. errors; 2) pooled OLS with standard errors assuming correlation over states; 3) pooled OLS with standard errors assuming general serial correlation in the error (to four lags) and correlation over states; 4) pooled OLS that assumes an AR(1) error and then gets standard errors that additionally permit correlation over states; 5) PFGLS with standard errors assuming an AR(1) error; and 6) PFGLS assuming an AR(1) error and correlation across states. In all cases of AR(1) error, we specialize to  $\rho_i = \rho$ .

```

. * Comparison of various pooled OLS and GLS estimators
. qui xtpcse lnc lnp lny lnpmin year, corr(ind) independent nmk
. estimates store OLS_iid
. qui xtpcse lnc lnp lny lnpmin year, corr(ind)
. estimates store OLS_cor
. qui xtscs lnc lnp lny lnpmin year, lag(4)
. estimates store OLS_DK
. qui xtpcse lnc lnp lny lnpmin year, corr(ar1)
. estimates store AR1_cor
. qui xtgls lnc lnp lny lnpmin year, corr(ar1) panels(iid)
. estimates store FGLSAR1
. qui xtgls lnc lnp lny lnpmin year, corr(ar1) panels(correlated)
. estimates store FGLSCAR
. estimates table OLS_iid OLS_cor OLS_DK AR1_cor FGLSAR1 FGLSCAR, b(%7.3f) se

```

Variable	OLS_iid	OLS_cor	OLS_DK	AR1_cor	FGLSAR1	FGLSCAR
lnp	-0.583	-0.583	-0.583	-0.266	-0.264	-0.330
	0.129	0.169	0.279	0.049	0.049	0.026
lny	0.365	0.365	0.365	0.398	0.397	0.407
	0.049	0.080	0.167	0.125	0.094	0.080
lnpmin	-0.027	-0.027	-0.027	0.069	0.070	0.036
	0.128	0.166	0.258	0.064	0.059	0.034
year	-0.033	-0.033	-0.033	-0.038	-0.038	-0.037
	0.004	0.006	0.012	0.010	0.007	0.006
_cons	6.930	6.930	6.930	5.115	5.100	5.393
	0.353	0.330	0.527	0.544	0.414	0.361

Legend: b/se

For pooled OLS with i.i.d. errors, the `nmk` option normalizes the VCE by  $N - K$  rather than  $N$ , so that output is exactly the same as that from `regress` with default standard errors. The same results could be obtained by using `xtgls` with the `corr(ind)` `panel(iid)` `nmk` options. Allowing correlation across states increases OLS standard errors by 30–50%. Additionally, allowing for serial correlation (`OLS_DK`) leads to another 50–100% increase in the standard errors. The fourth and fifth estimators control for at least an AR(1) error and yield roughly similar coefficients and standard errors. The final column results are similar to those given at the start of this section, where we used the more flexible `corr(psar1)` rather than `corr(ar1)`.

### 9.5.5 FE and RE models

As noted earlier, if there are few individuals and many time periods, individual-specific FE models can be fit with the least-squares dummy variable approach of including a set of dummy variables, here for each time period (rather than for each individual as in the short-panel case).

Alternatively, one can use the `xtregar` command. This model is the individual-effects model  $y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + u_{it}$ , with AR(1) error  $u_{it} = \rho u_{i,t-1} + \varepsilon_{it}$ . This is a better model of the error than the i.i.d. error model  $u_{it} = \varepsilon_{it}$  assumed in `xtreg`, so `xtregar` potentially will lead to more efficient parameter estimates.

The syntax of `xtregar` is similar to that for `xtreg`. The two key options are `fe` and `re`. The `fe` option treats  $\alpha_i$  as a fixed effect. Given an estimate of  $\hat{\rho}$ , we first transform to eliminate the effect of the AR(1) error, as described after (9.5), and then transform again (mean-difference) to eliminate the individual effect. The `re` option treats  $\alpha_i$  as a random effect.

We compare pooled OLS estimates, RE estimates using `xtreg` and `xtregar`, and FE estimates using `xtreg`, `xtregar`, and `xtscc`. Recall that `xtscc` calculates either the OLS or regular within estimator but then estimates the VCE assuming quite general error correlation over time and across states. We have

```

. * Comparison of various RE and FE estimators
. qui use mus209cigar, clear
. qui xtscc lnc lnp lny lnpmin, lag(4)
. estimates store OLS_DK
. qui xtreg lnc lnp lny lnpmin, fe
. estimates store FE_REG
. qui xtreg lnc lnp lny lnpmin, re
. estimates store RE_REG
. qui xtregar lnc lnp lny lnpmin, fe
. estimates store FE_REGAR
. qui xtregar lnc lnp lny lnpmin, re
. estimates store RE_REGAR
. qui xtscc lnc lnp lny lnpmin, fe lag(4)
. estimates store FE_DK
. estimates table OLS_DK FE_REG RE_REG FE_REGAR RE_REGAR FE_DK, b(%7.3f) se

```

Variable	OLS_DK	FE_REG	RE_REG	FE_RE^R	RE_RE^R	FE_DK
lnp	-0.611	-1.136	-1.110	-0.260	-0.282	-1.136
	0.438	0.101	0.102	0.049	0.052	0.162
lny	-0.027	-0.046	-0.045	-0.066	-0.074	-0.046
	0.027	0.011	0.011	0.064	0.026	0.020
lnpmin	-0.129	0.421	0.394	-0.010	-0.004	0.421
	0.346	0.101	0.102	0.057	0.060	0.172
_cons	8.357	8.462	8.459	6.537	6.708	8.462
	0.647	0.241	0.247	0.036	0.289	0.474

Legend: b/se

There are three distinctly different sets of coefficient estimates: those using pooled OLS, those using `xtreg` to obtain FE and RE estimators, and those using `xtregar` to obtain FE and RE estimators. The final set of estimates uses the `fe` option of the community-contributed `xtscc` command. This produces the standard within estimator but then finds standard errors that are robust to both spatial (across panels) and serial autocorrelation of the error.

## 9.5.6 Interactive effects

For panel data with many individuals and many time periods, [Bai \(2009\)](#) proposed a model with interactive effects that is a richer model than one

with additive individual-specific effects and time-specific effects.

The interactive effects model specifies

$$y_{it} = \mathbf{x}'_{it}\boldsymbol{\beta} + \boldsymbol{\lambda}'_i \mathbf{f}_t + \varepsilon_{it}$$

where  $\mathbf{f}_t$  is a vector of unobserved common effects or common time shocks and  $\boldsymbol{\lambda}_i$  is a corresponding vector of weights that vary at the individual level. In factor-analysis terminology,  $\mathbf{f}_t$  is a factor and  $\boldsymbol{\lambda}_i$  are factor loadings. The special case  $\boldsymbol{\lambda}'_i = (1, \alpha_i)$  and  $\mathbf{f}'_t = (1, \delta_t)$  yields  $\boldsymbol{\lambda}'_i \mathbf{f}_t = \alpha_i + \delta_t$ . The idiosyncratic error  $\varepsilon_{it}$  may be serially and cross-sectionally dependent.

Stacking observations over all time periods for a given individual yields model  $\mathbf{Y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{F}\boldsymbol{\Lambda}_i + \varepsilon_i$ . Then  $\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{F}}, \widehat{\boldsymbol{\Lambda}}$  minimize  $\sum_i (\mathbf{Y}_i - \mathbf{X}_i\boldsymbol{\beta} - \mathbf{F}\boldsymbol{\Lambda}_i)'(\mathbf{Y}_i - \mathbf{X}_i\boldsymbol{\beta} - \mathbf{F}\boldsymbol{\Lambda}_i)$ , where to ensure parameter identification, we normalize  $\mathbf{F}'\mathbf{F} = \mathbf{I}$  and  $\boldsymbol{\Lambda}'\boldsymbol{\Lambda}$  diagonal. Asymptotic theory requires  $N \rightarrow \infty$  and  $T \rightarrow \infty$ , with inference varying according to whether  $T/N \rightarrow 0$  or  $N/T \rightarrow 0$  or  $T/N \rightarrow c$  for some constant  $c > 0$ .

The interactive-effects model can be fit using the community-contributed command `regife` ([Gomez 2017](#)).

### 9.5.7 Separate regressions

The pooled regression specifies the same regression model for all individuals in all years. Instead, we could have a separate regression model for each individual unit:

$$y_{it} = \mathbf{x}'_{it}\boldsymbol{\beta}_i + u_{it}$$

This model has  $NK$  parameters, so inference is easiest for a long panel with a small  $N$ .

For example, suppose for the cigarette example we want to fit separate regressions for each state. Separate OLS regressions for each state can be obtained by using the `statsby` prefix with the `by(state)` option. We have

```
. * Run separate regressions for each state
. statsby, by(state) clear: regress lnc lnp lny lnpmin year
(running regress on estimation sample)

    Command: regress lnc lnp lny lnpmin year
        By: state
```

Statsby groups

		1		2		3		4		5
.....										

This leads to a dataset with 10 observations on `state` and the 5 regression coefficients. We have

```
. * Report regression coefficients for each state
. format _b* %9.2f
. list, clean
```

	state	_b_lnp	_b_lny	_b_lnp~n	_b_year	_b_cons
1.	1	-0.36	1.10	0.24	-0.08	2.10
2.	2	0.12	0.60	-0.45	-0.05	5.14
3.	3	-0.20	0.76	0.12	-0.05	2.72
4.	4	-0.52	-0.14	-0.21	-0.00	9.56
5.	5	-0.55	0.71	0.30	-0.07	4.76
6.	6	-0.11	0.21	-0.14	-0.02	6.20
7.	7	-0.43	-0.07	0.18	-0.03	9.14
8.	8	-0.26	0.89	0.08	-0.07	3.67
9.	9	-0.03	0.55	-0.36	-0.04	4.69
10.	10	-1.41	1.12	1.14	-0.08	2.70

In all states except one, sales decline as price rises, and in most states, sales increase with income.

One can also test for poolability, meaning to test whether the parameters are the same across states. In this example, there are  $5 \times 10 = 50$  parameters in the unrestricted model and 5 in the restricted pooled model, so there would be 45 restrictions to test.

## 9.5.8 Heterogeneous panels

[Pesaran and Smith \(1995\)](#) proposed aggregating the separate regressions to perform inference on  $E(\beta_i)$  by using the mean group (MG) estimator

$$\widehat{\boldsymbol{\beta}} = (1/N) \sum_{i=1}^N \widehat{\boldsymbol{\beta}}_i.$$

Consistency of the MG estimator requires that regressors be uncorrelated with the error. [Pesaran \(2006\)](#) relaxed this assumption by introducing interactive effects. The model is the same as the interactive effects model of [Bai \(2009\)](#), except that the parameters  $\beta_i$  vary across individuals. Pesaran shows that, for large  $N$  and  $T$ ,  $\beta_i$  can be consistently estimated by adding as regressors the time averages of  $y_{it}$  and  $\mathbf{x}_{it}$  and obtaining for each individual OLS estimates of the model

$$y_{it} = \mathbf{x}'_{it} \boldsymbol{\beta}_i + \gamma_i \bar{y}_t + \bar{\mathbf{x}}'_t \boldsymbol{\pi}_i + u_{it}$$

The common correlated estimator (CCE) of  $E(\boldsymbol{\beta}_i)$  is then

$$\widehat{\boldsymbol{\beta}} = (1/N) \sum_{i=1}^N \widehat{\boldsymbol{\beta}}_i.$$

The CCE estimator treats the factor variables  $\mathbf{f}_t$  as nuisance parameters, but in some applications estimates of  $\mathbf{f}_t$  are of intrinsic interest. The augmented MG estimator of [Eberhardt and Teal \(2010\)](#) extends the CCE estimator by providing estimates of  $\mathbf{f}_t$ .

[Pesaran and Smith \(1995\)](#) proposed a pooled mean group estimator that extends the CCE estimator to dynamic models with regressors that potentially follow a unit-root process.

The community-contributed `xtpmg` command ([Blackburne and Frank 2007](#)) implements the MG and pooled mean group estimators. The community-contributed `xtmg` command ([Eberhardt 2012](#)) implements the MG, CCE, and augmented MG estimators.

### 9.5.9 Unit roots and cointegration

For time series that are nonstationary because of unit roots, estimators no longer have an asymptotic normal distribution as  $T \rightarrow \infty$ ; see, for example, [Greene \(2018, chap. 21\)](#). Similar nonstandard distributions arise for panel

data with unit roots when asymptotic theory relies on  $T \rightarrow \infty$ ; see [Baltagi \(2021, chap. 12\)](#).

If  $N$  is small, say,  $N < 10$ , then seemingly unrelated equations methods can be used. When  $N$  is large, the panel aspect becomes more important. Complications include the need to control for cross-sectional unobserved heterogeneity when  $N$  is large, asymptotic theory that can vary with exactly how  $N$  and  $T$  both go to infinity, and the possibility of cross-sectional dependence. At the same time, statistics that have nonnormal distributions for a single time series can be averaged over cross-sections to obtain statistics with a normal distribution.

Unit-root tests can have low power. Panel data may increase the power because of now having time series for several cross-sections. The unit-root tests can also be of interest per se, such as testing purchasing power parity, as well as being relevant to consequent considerations of cointegration. A dynamic model with cross-sectional heterogeneity is

$$y_{it} = \rho_i y_{i,t-1} + \phi_{i1} \Delta y_{i,t-1} + \cdots + \phi_{ip_i} \Delta y_{i,t-p_i} + \mathbf{z}'_{it} \boldsymbol{\gamma}_i + u_{it}$$

where lagged changes are introduced so that  $u_{it}$  is i.i.d. Examples of  $\mathbf{z}_{it}$  include individual effects, with  $\mathbf{z}_{it} = (1)$ ; individual effects and individual time trends, with  $\mathbf{z}_{it} = (1 \ t)'$ , and  $\boldsymbol{\gamma}_i = \boldsymbol{\gamma}$  in the case of homogeneity. A unit-root test is a test of  $H_0: \rho_1 = \cdots = \rho_N = 1$ . [Levin, Lin, and Chu \(2002\)](#) proposed a test against the alternative of homogeneity,

$H_a: \rho_1 = \cdots = \rho_N = \rho < 1$ , that is based on pooled OLS estimation using specific first-step pooled residuals, where in both steps homogeneity ( $\rho_i = \rho$  and  $\phi_{ik} = \phi_k$ ) is imposed.

A number of different unit-root tests with panel data have been proposed, including tests that relax the assumption of independence across individuals to allow correlation. The `xtunitroot` command provides six different panel unit-root tests.

As in the case of a single time series, cointegration tests are used to ensure that statistical relationships between trending variables are not

spurious. A quite general cointegrated panel model is

$$y_{it} = \mathbf{x}'_{it}\boldsymbol{\beta}_i + \mathbf{z}'_{it}\boldsymbol{\gamma}_i + u_{it}$$
$$\mathbf{x}_{it} = \mathbf{x}_{i,t-1} + \varepsilon_{it}$$

where  $\mathbf{z}_{it}$  is deterministic and can include individual effects and time trends and  $\mathbf{x}_{it}$  are (co)integrated regressors.

Most tests of cointegration are based on the OLS residuals  $\hat{u}_{it}$ , but the unit-root tests cannot be directly applied if  $\text{Cov}(u_{it}, \varepsilon_{it}) \neq 0$ , as is likely. The `xtcointtest` command provides three different panel cointegration tests.

Single-equation estimators have been proposed that generalize to panels fully modified OLS and dynamic OLS, and Johanssen's system approach has also been generalized to panels.

For further details, see the books by [Baltagi \(2021\)](#) and [Pesaran \(2015\)](#).

## 9.6 Additional resources

The major reference is [XT] *Stata Longitudinal/Panel-Data Reference Manual*, especially [XT] **xtivreg**, [XT] **xthtaylor**, and [XT] **xtabond**. The community-contributed `xtivreg2` command ([Schaffer 2005](#)) provides a broad range of IV-related estimators and associated tests. For estimation with long panels, a useful Stata community-contributed command is `xtscc`, as well as several others mentioned in section [9.5. Fernández-Val and Weidner \(2018\)](#) survey finite- $T$  and finite- $N$  bias for fixed-effects estimators. [Karabiyik, Palm, and Urbain \(2019\)](#) survey panel models with cross-sectional dependence. [Chiang, Hansen, and Sasaki \(2021\)](#) propose an extension to two-way cluster-robust standard errors that is also robust against common time effects.

Many of the topics in this chapter appear in more specialized books on panel data, notably, [Arellano \(2003\)](#), [Baltagi \(2021\)](#), [Hsiao \(2014\)](#), [Lee \(2002\)](#), and [Pesaran \(2015\)](#). Cameron and Trivedi ([2005](#)) present many of the methods in this chapter.

## 9.7 Exercises

1. For the model and data of section [9.2](#), obtain the panel IV estimator in the FE model by applying the `ivregress` command to the mean-differenced model with a mean-differenced instrument. Hint: For example, for variable  $x$ , type `by id: egen avex = mean(x)` followed by `summarize x` and then generate `mdx = x-avex+r(mean)`. Verify that you get the same estimated coefficients as you would with `xtivreg, fe`.
2. For the model and data of section [9.4](#), use the `xtdpdsys` command given in section [9.4.6](#), and then perform specification tests using the `estat abond` and `estat sargan` commands. Use `xtdpd` at the end of section [9.4.8](#), and compare the results with those from `xtdpdsys`. Is this what you expect, given the results from the preceding specification tests?
3. Consider the model and data of section [9.4](#), except consider the case of just one lagged dependent variable. Throughout, estimate the parameters of the models with the `noconstant` option. Consider estimation of the dynamic model  $y_{it} = \alpha_i + \gamma y_{it-1} + \varepsilon_{it}$ , when  $T = 7$ , where  $\varepsilon_{it}$  are serially uncorrelated. Explain why OLS estimation of the transformed model  $\Delta y_{it} = \gamma_1 \Delta y_{it-1} + \Delta \varepsilon_{it}$ ,  $t = 2, \dots, 7$ , leads to inconsistent estimation of  $\gamma_1$ . Propose an instrumental-variable estimator of the preceding model where there is just one instrument. Implement this just-identified IV estimator using the data on `lwage` and the `ivregress` command. Obtain cluster-robust standard errors. Compare with OLS estimates of the differenced model.
4. Continue with the model of the previous question. Consider the Arellano–Bond estimator. For each time period, state what instruments are used by the `estat abond` command. Perform the Arellano–Bond estimator using the data on `lwage`. Obtain the one-step estimator with robust standard errors. Obtain the two-step estimator with robust standard errors. Compare the estimates and their standard errors. Is there an efficiency gain compared with your answer in the previous question? Use the `estat abond` command to test whether the errors  $\varepsilon_{it}$  are serially uncorrelated. Use the `estat sargan` command to test whether the model is correctly specified.



# **Chapter 10**

## **Introduction to nonlinear regression**

## 10.1 Introduction

This chapter provides a brief introduction to nonlinear regression methods that are presented in much more detail in subsequent chapters.

We focus on the most commonly used nonlinear-in-parameters models in microeconomics, the probit and logit models for binary outcomes that take only two values. Examples include whether a commuter uses a car or uses other means of transport, whether a person is employed, and whether a person visits a doctor.

Unlike the case for linear models, the marginal effect (ME) of a change in a regressor is no longer simply the associated slope parameter. This can make direct interpretation of parameter estimates difficult, so interpretation of results relies on estimated MES. And estimation uses iterative numerical methods because there is no closed-form solution for parameter estimates.

For standard nonlinear models, simply changing the command from `regress y x` to `probit y x`, for example, leads to nonlinear estimation and regression output that looks essentially the same as the output from `regress`. The *p*-values and confidence intervals given are only approximate, however, unless the sample size is very large. MES can be computed using the `margins` postestimation command.

## 10.2 Binary outcome models

We consider probit regression and logit regression for whether a person visits a doctor. There is no need to first read the separate chapter 17 on binary outcome models because any necessary background is provided here.

### 10.2.1 Doctor visit example

Data on office-based physician visits by persons in the United States aged 25–64 years come from the 2002 Medical Expenditure Panel Survey. The sample is the same as that used by [Deb, Munkin, and Trivedi \(2006\)](#). It excludes those receiving public insurance (Medicare and Medicaid) and is restricted to those working in the private sector but not self-employed.

The dependent variable (`visit`) is a binary variable equal to 1 if the person visited a doctor at least once and equal to 0 if the person did not. The regressors used here are restricted to health insurance status (`private`), health status (`chronic`) and two socioeconomic characteristics (`female` and `income`) to keep Stata output short. We have

Variable name	Storage type	Display format	Value label	Variable label
visit	float	%9.0g	= 1 if doctor visit	
private	byte	%8.0g	Private insurance	
chronic	byte	%8.0g	Chronic condition	
female	byte	%8.0g	Female	
income	float	%9.0g	Income in \$ / 1000	

Summary statistics for these variables follow.

```
. * Summary of key variables
. summarize visit private chronic female income
```

Variable	Obs	Mean	Std. dev.	Min	Max
visit	4,412	.6359927	.4812052	0	1
private	4,412	.7853581	.4106202	0	1
chronic	4,412	.3263826	.4689423	0	1
female	4,412	.4718948	.4992661	0	1
income	4,412	34.34018	29.03987	-49.999	280.777

From the summary statistics, 64% of the sample see a doctor at least once (`visit=1`), 79% have private health insurance, 33% have a chronic condition, 47% are female, and average annual income is \$34,340. We use all the sample, including the three people who from command `tabulate income` (output not given) have negative income.

### 10.2.2 Probit and logit model definition

For binary outcome models, the dependent variable  $y$  takes just two values, set to 1 and 0 for simplicity. It is then clear that the distribution of  $y$  is the Bernoulli, the same as that for a coin toss. If the probability that  $y = 1$  is  $p$ , then the probability that  $y = 0$  is necessarily  $1 - p$ .

Binary outcome regression models allow  $p$  to vary with the regressors  $\mathbf{x}$ . A linear model  $p = \mathbf{x}'\boldsymbol{\beta}$  does not restrict the probability to lie between zero and one. Better models set  $p = F(\mathbf{x}'\boldsymbol{\beta})$ , where  $F(\cdot)$  is a known function with the property that  $0 < F(\cdot) < 1$ . Any cumulative distribution function (c.d.f.)  $F(z)$  for variable  $z$  continuous on  $(-\infty, \infty)$  has this desired property. The probit and logit models differ in this specified function  $F(\cdot)$ .

The probit regression model specifies that

$$\Pr(y_i = 1 | \mathbf{x}_i) = \Phi(\mathbf{x}'_i \boldsymbol{\beta})$$

where  $\Phi(\cdot)$  is the standard normal c.d.f., defined as  

$$\Phi(z) = \int_{-\infty}^z (2\pi)^{-1/2} e^{-t^2/2} dt.$$

The logit regression model specifies

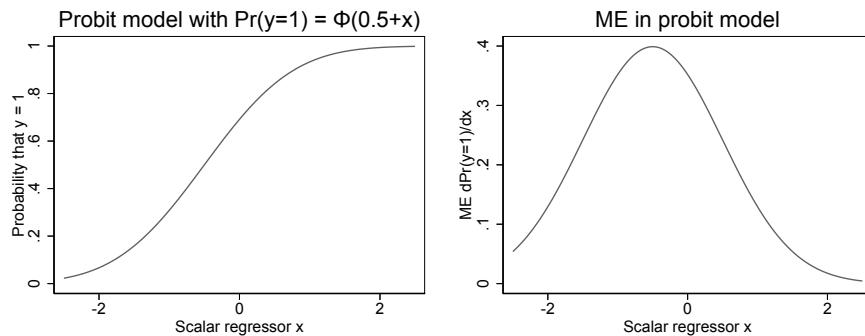
$$\Pr(y_i = 1|\mathbf{x}_i) = \Lambda(\mathbf{x}'_i \boldsymbol{\beta}) \quad (10.1)$$

where  $\Lambda(\cdot)$  is the logistic c.d.f. and  $\Lambda(z) = e^z/(1 + e^z)$ .

As detailed in section [10.4](#), logit and probit models give different parameter estimates because of the different choice of function. But the two models produce very similar model fit, predicted probabilities, and MES. Applied economists most often use the probit model, while in other areas of applied statistics such as biostatistics, the logit model is most frequently used.

The probit and logit models are nonlinear functions of the regressors, so the ME on  $\Pr(y = 1|\mathbf{x})$  of a change in the  $j$ th regressor is no longer simply  $\beta_j$ ; it additionally depends on the value of the regressors  $\mathbf{x}$ . For the probit model, for example, the ME equals  $\beta_j \phi(\mathbf{x}' \boldsymbol{\beta})$ , where  $\phi(\cdot)$  is the standard normal density.

Consider the simple probit model with a scalar regressor  $x$  and  $\Pr(y = 1|x) = \Phi(0.5 + x)$ . The first panel of figure [10.1](#) plots  $\Pr(y = 1|x)$  against  $x$ . The relationship is clearly nonlinear. The second panel of figure [10.1](#) plots the ME on  $\Pr(y = 1|x)$  of a change in  $x$ . The ME clearly varies with the value of  $x$ . Computation of MES using the `margins` command is detailed below.



**Figure 10.1.** Probit model and associated ME

The probit and logit estimators are obtained by maximum likelihood methods, detailed in section 17.3. Consistency requires that the function  $\Pr(y = 1|\mathbf{x})$  be correctly specified. Once this function is correctly specified, the entire distribution is correctly specified because  $y$  takes only two values and  $\Pr(y = 0|\mathbf{x}) = 1 - \Pr(y = 1|\mathbf{x})$  necessarily because probabilities sum to 1.

## 10.3 Probit model

We focus on probit regression. Qualitatively similar analysis holds for logit regression, presented in section [10.5](#).

### 10.3.1 The probit command

The probit ML estimator is obtained using the `probit` command. The syntax of the command is

```
probit depvar [indepvars] [if] [in] [weight] [, options]
```

This syntax is the same as that for command `regress`. Usually, there is no need to use any of the command options, aside from the `vce(vcetype)` option.

### 10.3.2 Probit estimation results

We use the `probit` command, with the `vce(robust)` option that provides heteroskedastic-robust standard errors; see section [10.3.3](#) for discussion of which standard errors to use.

This yields the following results for whether a person visited the doctor.

```

. * Probit regression (command probit) with robust standard errors
. probit visit private chronic female income, vce(robust)

Iteration 0: log pseudolikelihood = -2892.9
Iteration 1: log pseudolikelihood = -2337.6553
Iteration 2: log pseudolikelihood = -2331.8213
Iteration 3: log pseudolikelihood = -2331.8084
Iteration 4: log pseudolikelihood = -2331.8084

Probit regression                                         Number of obs = 4,412
                                                               Wald chi2(4) = 910.77
                                                               Prob > chi2 = 0.0000
                                                               Pseudo R2 = 0.1940

Log pseudolikelihood = -2331.8084

```

visit	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	.7663244	.0528282	14.51	0.000	.6627832	.8698657
chronic	1.064481	.0511394	20.82	0.000	.9642499	1.164713
female	.5529806	.0434216	12.74	0.000	.4678759	.6380854
income	.0056173	.0008671	6.48	0.000	.0039178	.0073169
_cons	-.9594421	.0525925	-18.24	0.000	-1.062521	-.8563626

The output begins with an iteration log because the estimator is obtained numerically using an iterative procedure presented in section 16.2. In this case, four iterations are needed. Each iteration increases the log-likelihood function, as desired, and iterations cease when there is little change in the log-likelihood function.

The remaining output from `probit` is remarkably similar to that for `regress`. The 4 regressors are jointly statistically significant at 5% because the Wald  $\chi^2(4)$  statistic has  $p = 0.00 < 0.05$ . This test is chi-squared distributed rather than  $F$  distributed because it relies on asymptotic theory. The pseudo- $R^2$ , discussed with other model diagnostics in section 13.8, does not have the same interpretation as  $R^2$  in the linear regression model. There is no analysis-of-variance table because this table is appropriate only for linear least squares with independent and identically distributed errors.

The remaining output indicates that all regressors are individually statistically significant at level 0.05 because all  $p$ -values are less than 0.05. For each regressor, the output presents in turn the following:

Coefficients	$\hat{\beta}_j$
Standard errors	$s_{\hat{\beta}_j}$
$z$ statistics	$z_j = \hat{\beta}_j / s_{\hat{\beta}_j}$
$p$ -values	$p_j = \Pr \{  z_j  > 0   z_j \sim \mathcal{N}[0, 1] \}$
95% confidence intervals	$\hat{\beta}_j \pm 1.96 \times s_{\hat{\beta}_j}$

The  $z$  statistics and  $p$ -values are computed using the standard normal distribution, rather than the  $t$  distribution with  $N - K$  degrees of freedom, because they are based on asymptotic normality. The  $p$ -values are for a two-sided test of whether  $\beta_j = 0$ . For a one-sided test of  $H_0: \beta_j \leq 0$  against  $\beta_j > 0$ , the  $p$ -value is half that reported in the table, provided that  $z_j > 0$ . For a one-sided test of  $H_0: \beta_j \geq 0$  against  $\beta_j < 0$ , the  $p$ -value is half that reported in the table, provided  $z_j < 0$ .

A nonlinear model raises a new issue of interpretation of slope coefficients  $\beta_j$ . For example, what does the value 0.0056 for the coefficient of `income` mean? We address this important issue in detail in section [10.4](#).

### 10.3.3 Standard error computation

As for linear regression, there are several ways to obtain standard errors, depending on the types of data being analyzed. Complete details, with formulas, are given in section [13.4](#). A brief summary is given here.

Default standard errors are based on the assumption that a model is correctly specified. With real-world data, however, even a good model is unlikely to be exactly correctly specified. And standard errors are needed not only for estimates of model parameters but also for subsequent statistics that are calculated such as MES.

For data that are independent across observations, it is standard to instead obtain heteroskedastic–robust standard errors, which for most commands are obtained using the `vce(robust)` option of a cross-sectional estimation command. With data that are clustered, with observations correlated if in the same cluster but independent if in different clusters, it is standard to obtain cluster–robust standard errors. These can be obtained by

using the `vce(cluster clustvar)` option and, more simply by using the `vce(robust)` for commands such as `xt` commands, which are for clustered data.

The preceding `probit` results used the `vce(robust)` option. Using default standard errors instead, we obtain

```
. * Probit regression (command probit) with default standard errors
. probit visit private chronic female income, nolog
Probit regression
Number of obs = 4,412
LR chi2(4) = 1122.18
Prob > chi2 = 0.0000
Pseudo R2 = 0.1940
Log likelihood = -2331.8084
```

visit	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.7663244	.0529517	14.47	0.000	.662541 .8701079
chronic	1.064481	.0510181	20.86	0.000	.9644878 1.164475
female	.5529806	.0433727	12.75	0.000	.4679716 .6379897
income	.0056173	.0008175	6.87	0.000	.0040151 .0072196
_cons	-.9594421	.0527427	-18.19	0.000	-1.062816 -.8560683

The standard errors differ by at most 6.5% for `income`. For some commands, there can be little difference between default and heteroskedastic–robust, whereas for others, notably for `poisson`, there can be a very substantial difference.

For data that are clustered, such as in short panels of independent individuals, or cross-sectional data with natural groupings, such as villages, one must use the option `vce(cluster clustvar)` to obtain cluster–robust standard errors. If observations are correlated within cluster, these correct standard errors can be much larger than the incorrect default standard errors or heteroskedastic–robust standard errors. For cluster–robust standard errors, the number of clusters should be large; see section [3.4.6](#).

Stata estimation commands with the `vce(bootstrap)` option provide standard errors using the bootstrap. The default is a paired bootstrap, which will be discussed in more detail in section [12.8.1](#) for the linear regression model, that assumes independent observations. It is asymptotically equivalent to computing heteroskedastic–robust standard errors, provided the number of bootstraps is large. Similarly, a cluster bootstrap that assumes

independence across clusters but not within cluster (the `vce(bootstrap, cluster(clustvar))` option) is asymptotically equivalent to computing cluster-robust standard errors, provided the number of bootstraps is large.

Bootstrap methods and the jackknife method (the `vce(jackknife)` option) are detailed in chapter 12. In that chapter, we additionally consider a different use of the bootstrap to implement a more refined asymptotic theory that can lead to  $t$  statistics with better size properties and confidence intervals with better coverage in finite samples.

#### 10.3.4 Postestimation commands

The `ereturn list` command provides a list of what estimation results are stored in `e()`; see section 1.6.2 for details following the `regress` command. Stored results include regression coefficients in `e(b)` and the estimated VCE in `e(V)`.

Standard postestimation commands available after most estimation commands, including nonlinear model commands such as the `probit` command, have already been given in table 3.1. Many of these postestimation commands have already been used in preceding chapters. In this chapter, we use the `predict` and `margins` commands.

To find specific postestimation commands available after command `probit`, for example, see [R] **probit postestimation** or use command `help probit postestimation`. This lists additional postestimation commands that are specific to the `probit` command: `estat classification`, `estat gof`, `lroc`, and `lsens`.

#### 10.3.5 Prediction

The `predict` command with the `pr` option, the default option after the `probit` command, computes for each observation  $\Phi(\mathbf{x}_i'\hat{\boldsymbol{\beta}})$ , the predicted probability that  $y = 1$ .

We obtain the following predicted probabilities:

```

. * Predicted probabilities from probit
. qui probit visit private chronic female income, vce(robust)
. predict phatprobit, pr
. summarize visit phatprobit

```

Variable	Obs	Mean	Std. dev.	Min	Max
visit	4,412	.6359927	.4812052	0	1
phatprobit	4,412	.6356915	.2316533	.168668	.992829

The predicted probabilities range from 0.169 to 0.993. The average predicted probability of 0.6357 is very close to the sample proportion 0.6360 of individuals who visited a doctor.

## 10.4 MEs and coefficient interpretation

An ME or a partial effect measures the effect of a change in one of the regressors, say,  $x_j$ . For nonlinear models, such as probit and logit, there are remarkably many different methods for calculating MES.

For the probit model, the ME of interest is that for the conditional probability that the event of interest happens, and the discussion below focuses on MES for  $\Pr(y = 1|\mathbf{x})$ . The methods carry over immediately to the more common case of computing MES for the conditional mean because in binary outcome models,  $E(y|\mathbf{x}) = \Pr(y = 1|\mathbf{x})$ .

While we focus on nonlinear models here, the discussion overlaps considerably with the methods for linear regression presented in section [4.5](#).

### 10.4.1 Calculus method and finite-difference method

Using calculus for the probit model, we see the ME of the  $j$ th regressor is

$$\text{ME}_j = \frac{\partial \Pr(y = 1|\mathbf{x})}{\partial x_j} = \phi(\mathbf{x}'\boldsymbol{\beta})\beta_j \quad (10.2)$$

This ME is not simply the relevant parameter  $\beta_j$ , and it varies with the point of evaluation  $\mathbf{x}$ .

Calculus methods are not always appropriate. In particular, for an indicator variable, say,  $d$ , the relevant ME is the discrete change in the conditional mean when  $d$  changes from 0 to 1.

Let  $\mathbf{x} = [\mathbf{z} \ d]$ , where  $\mathbf{z}$  denotes all regressors other than the  $j$ th, which is an indicator variable  $d$ , and let  $\Pr(y = 1|\mathbf{z}, d) = \Phi(\mathbf{z}'\boldsymbol{\beta}_1 + \beta_2d)$ . Then the finite-difference method yields ME

$$\begin{aligned} \text{ME}_j &= \Pr(y = 1|\mathbf{z}, d = 1) - \Pr(y = 1|\mathbf{z}, d = 0) \\ &= \Phi(\mathbf{z}'\boldsymbol{\beta}_1 + \beta_2) - \Phi(\mathbf{z}'\boldsymbol{\beta}_1) \end{aligned}$$

## 10.4.2 Average marginal effect, ME at mean, and ME at a representative value

The ME varies with the point of evaluation  $x$ . Three common choices of evaluation are 1) at sample values and then average; 2) at the sample mean of the regressors; and 3) at representative values of the regressors.

We use the following acronyms, where the first two follow [Bartus \(2005\)](#).

AME	Average ME	Average of individual MEs
MEM	Marginal effect at mean	ME at $x = \bar{x}$
MER	Marginal effect at a representative value	ME at $x = x^*$

The MEM and MER can differ substantially from the AME in nonlinear models.

The ME averaged over individuals, the AME, is most commonly reported. The default is to compute a within sample average. A population AME can be obtained using the `weight` option of the `margins` command, provided that sampling weights are available.

Sometimes, interest lies instead in the ME for the average individual; then the MEM is computed. And if interest lies in the ME for a particular representative individual, then the MER is computed.

## 10.4.3 AME in treatment-effects models

In the simplest treatment-effects examples, the treatment is a variable that takes one of two values according to whether an individual receives treatment. Interest lies in measuring the average treatment effect (ATE), the average difference across individuals in the outcome according to whether treated or not.

In the current context of a parametric model, the treatment variable can be included as a binary regressor, and the AME for that regressor is the ATE.

The treatment-effects literature presented in chapters 24 and 25 instead presents methods that focus on estimating the ATE, or the closely related ATE

on the treated, under assumptions that can be weaker than those considered here. These assumptions do not necessarily require obtaining an unbiased or consistent estimate of the ME for each individual before averaging. For example, in a randomized experiment, we observe each individual's outcome in only one of the two states of treatment or no treatment, so we cannot estimate the ME of treatment for a given individual. But if assignment is random, we can obtain the ATE in this context, as the difference in the average outcome of the two groups.

#### 10.4.4 The margins and margins, dydx commands

The three MES measures can be computed using the `margins` postestimation command.

The syntax of the command of the `margins` command is

```
margins [ marginlist ] [ if ] [ in ] [ weight ] [ , response_options options ]
```

where *marginlist* is a list of factor variables or interactions that appear in the current estimation results, *response\_options* specify the particular quantity to be computed, and *options* include particular values of regressors at which computation occurs. The `margins` command without any options computes the sample average value of the default quantity computed by the `predict` command.

MES are computed using `dydx()`. The arguments can be a list of regressors, while `dydx(*)` computes the ME for all regressors. The default is to compute the AME. The option `atmeans` computes the MEM, and option `at()` computes the MER at specified values of the regressors.

#### 10.4.5 Probit model application

We compute the average of MES across individuals (the AME) for the previously fit probit model.

```

. * AMEs using calculus method
. qui probit visit private chronic female income, vce(robust)
. margins, dydx(*)
```

Average marginal effects Number of obs = 4,412  
 Model VCE: Robust  
 Expression: Pr(visit), predict()  
 dy/dx wrt: private chronic female income

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
private	.2291266	.0146652	15.62	0.000	.2003833	.2578699
chronic	.3182738	.0131955	24.12	0.000	.292411	.3441366
female	.165338	.0122877	13.46	0.000	.1412546	.1894215
income	.0016796	.0002555	6.57	0.000	.0011787	.0021804

When averaged across individuals, the probability of visiting a doctor is 0.229 higher, or 22.9 percentage points higher, for someone with private insurance. And a \$1,000 increase in income, a one-unit change in variable income, is associated with a 0.00168 increase in the probability of visiting a doctor. The standard errors and associated  $z$  statistics and  $p$ -values are based on heteroskedastic-robust standard errors because the immediately preceding probit command used the vce(robust) option.

These AMES are computed using calculus methods. For binary regressors private, chronic, and female, it is more natural to use the finite-difference method. This can be done by fitting the model using the factor variable prefix i. for the discrete-valued regressors.

For the probit model using factor variables, we obtain

```

. * AMEs using finite-difference method
. qui probit visit i.private i.chronic i.female income, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Pr(visit), predict()
dy/dx wrt:  1.private 1.chronic 1.female income

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	.2508057	.0175548	14.29	0.000	.216399	.2852123
1.chronic	.313282	.0128372	24.40	0.000	.2881215	.3384425
1.female	.1693009	.0130795	12.94	0.000	.1436654	.1949363
income	.0016796	.0002555	6.57	0.000	.0011787	.0021804

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

There is some change in the estimated AME for the binary regressors, as great as from 0.229 to 0.251 for regressor `private`. There is no change for the continuous regressor `income`.

Factor variables are especially useful for obtaining MES in models with interacted regressors. For example, suppose private health insurance status is interacted with variable income. Then,

```
. * AMEs with interacted regressors
. probit visit i.private##c.income i.chronic i.female, vce(robust) nolog noheader
```

visit	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.private income	.7278849 .0037553	.0800428 .0028952	9.09 1.30	0.000 0.195	.5710039 -.0019192	.8847659 .0094297
private# c.income 1	.0020059	.0030257	0.66	0.507	-.0039244	.0079363
1.chronic 1.female _cons	1.065191 .552433 -.9263058	.0511475 .0434335 .073073	20.83 12.72 -12.68	0.000 0.000 0.000	.9649438 .4673049 -1.069526	1.165438 .6375611 -.7830854

```
. margins, dydx(*)
Average marginal effects
Model VCE: Robust
Number of obs = 4,412
Expression: Pr(visit), predict()
dy/dx wrt: 1.private income 1.chronic 1.female
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private income	.2609297 .0015848	.0228076 .0002874	11.44 5.51	0.000 0.000	.2162276 .0010214	.3056317 .0021482
1.chronic 1.female	.3134803 .1691342	.0128384 .0130861	24.42 12.92	0.000 0.000	.2883176 .1434859	.338643 .1947824

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

The regressors include income, private, and income×private. The margins command obtains the AME for income and the AME for private health insurance, allowing for their interaction. The AMES are very close to those for the model without interactions because the interaction term had very little explanatory power.

The standard errors that are reported above hold the regressor fixed. The option vce(unconditional) of the margins command additionally allows for variation due to sampling of the regressor; see section 13.7.9. In the preceding examples, this leads to less than a 1% increase in the standard errors of the AMES.

## 10.4.6 Simple interpretations of coefficients

In a single-index model, the regressors  $\mathbf{x}$  enter as a function of the linear combination  $\mathbf{x}'\boldsymbol{\beta}$ . The probit and logit models, like many other nonlinear models, are of single-index form. Coefficient interpretation is simplified in such models.

We focus on the probit model, with function  $\Phi(\mathbf{x}'\boldsymbol{\beta})$ . Then from (10.2) the ME is  $\text{ME}_j = \phi(\mathbf{x}'\boldsymbol{\beta})\beta_j$ , where  $\phi(\cdot)$  is the standard normal density. Because  $\phi(\cdot) \geq 0$  always, it follows that the sign of the ME equals the sign of  $\beta_j$ . For example, if  $\beta_j > 0$ , then an increase in  $x_j$  is associated with an increase in  $\Pr(y = 1)$ .

The ratio of MES for two different regressors equals the ratio of the corresponding parameters because

$$\frac{\text{ME}_j}{\text{ME}_k} = \frac{\phi(\mathbf{x}'\boldsymbol{\beta})\beta_j}{\phi(\mathbf{x}'\boldsymbol{\beta})\beta_k} = \frac{\beta_j}{\beta_k}$$

Therefore, if one coefficient is twice as big as another, then so too is the ME. This property applies to most commonly used nonlinear regression models, aside from multinomial models, where regressors appear as a linear combination  $\mathbf{x}'\boldsymbol{\beta}$ .

For example, from section 10.3.2, regressor `private` has coefficient 0.766, and regressor `chronic` has coefficient 1.064. The effects for both regressors are positive because the coefficients are positive and  $\phi(\mathbf{x}'\boldsymbol{\beta})$  is positive. And having a chronic condition is associated with a 1.4 times bigger change in doctor visits than having private insurance ( $1.064/0.766 = 1.4$ ).

Additional interpretation of coefficients can be possible for specific single-index models. For example, for the probit model,  $\text{ME}_j = \phi(\mathbf{x}'\boldsymbol{\beta})\beta_j \leq 0.4\beta_j$  as  $\phi(z)$  takes a maximum value of  $1/\sqrt{2\pi} \simeq 0.399$  at  $z = 0$ .

#### 10.4.7 Comparison with linear least squares

What if we fit the model by OLS regression rather than probit regression?

This alternative model is called the linear probability model. It specifies that

$$\Pr(y_i = 1 | \mathbf{x}_i) = \mathbf{x}'_i \boldsymbol{\beta}$$

which does not restrict the probability that  $y = 1$  be in the  $(0, 1)$  interval. And it is implicitly based on an underlying normal distribution for  $y$ , whereas the data are clearly Bernoulli distributed.

The `regress` command yields

```
. * OLS regression (command regress) for linear probability model
. regress visit private chronic female income, vce(robust) noheader
```

visit	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
private	.2651354	.0171919	15.42	0.000	.2314307	.2988402
chronic	.3059204	.0126069	24.27	0.000	.2812046	.3306361
female	.170008	.0131967	12.88	0.000	.1441359	.19588
income	.0016152	.000227	7.11	0.000	.0011701	.0020603
_cons	.1922274	.0154689	12.43	0.000	.1619006	.2225542

```
. predict phatols, xb
. summarize visit phatols
```

Variable	Obs	Mean	Std. dev.	Min	Max
visit	4,412	.6359927	.4812052	0	1
phatols	4,412	.6359927	.2290251	.1922274	1.216793

The slope coefficient estimates are approximately one-third the size of the corresponding probit estimates given in section [10.3.2](#). At the same time, the OLS slope coefficients are equal to the MES because the model is linear. These MES are quite close to the probit model AMES given in section [10.4.5](#). The biggest difference is 0.265 versus 0.229 (using calculus methods) or 0.251 (using the finite-difference method) for variable `private`.

In this example, one observation had a predicted probability as high as 1.217, so OLS regression did lead to predicted probabilities outside the  $(0, 1)$  interval. More generally, OLS estimation may be a useful first step, but aside

from the special case of a fully saturated model, it is best to use the probit or logit model.

## 10.5 Logit model

The logit model defined in (10.1) specifies  $\Pr(y_i = 1 | \mathbf{x}_i)$  to equal  $\Lambda(\mathbf{x}'_i \boldsymbol{\beta})$ , where  $\Lambda(z) = e^z / (1 + e^z)$  rather than  $\Phi(\mathbf{x}'_i \boldsymbol{\beta})$ .

The functions  $\Lambda(\cdot)$  and  $\Phi(\cdot)$  not only are different functions but also are scaled quite differently. For example, while  $\Phi(0) = \Lambda(0) = 0.5$  and both functions are symmetric about 0,  $\Phi(1) \simeq 0.84 \neq \Lambda(1) \simeq 0.73$ . In fact, it can be shown that  $\Phi(z) \simeq \Lambda(1.7z)$ , so we might expect that  $\Phi(\mathbf{x}' \boldsymbol{\beta}_{\text{probit}}) \simeq \Lambda(1.7\mathbf{x}' \boldsymbol{\beta}_{\text{probit}})$ , leading to logit coefficients that will be approximately 1.7 times the probit coefficients.

The `logit` command has syntax similar to that given for the probit model in section 10.3.1. Logit regression in the current application yields

```
. * Logit regression (command logit)
. logit visit private chronic female income, vce(robust)

Iteration 0:  log pseudolikelihood = -2892.9
Iteration 1:  log pseudolikelihood = -2349.2911
Iteration 2:  log pseudolikelihood = -2332.1534
Iteration 3:  log pseudolikelihood = -2332.1197
Iteration 4:  log pseudolikelihood = -2332.1197

Logistic regression                                         Number of obs = 4,412
                                                               Wald chi2(4) = 798.75
                                                               Prob > chi2 = 0.0000
                                                               Pseudo R2 = 0.1938

Log pseudolikelihood = -2332.1197
```

visit	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	1.27266	.0896929	14.19	0.000	1.096866	1.448455
chronic	1.832121	.092782	19.75	0.000	1.650271	2.01397
female	.9280678	.0737619	12.58	0.000	.7834971	1.072639
income	.0095378	.0015245	6.26	0.000	.0065497	.0125258
_cons	-1.607674	.0907181	-17.72	0.000	-1.785478	-1.42987

The logit slope coefficients are indeed approximately 1.7 times the probit slope coefficients obtained in section 10.3.2. At the same time, the *t*-ratios are within 5% of each other.

The ME in the logit model obtained using calculus methods is

$$ME_j = [\Lambda(\mathbf{x}'\boldsymbol{\beta}) \times \{1 - \Lambda(\mathbf{x}'\boldsymbol{\beta})\}]\beta_j$$

Because  $0 \leq \Lambda(\cdot) \times \{1 - \Lambda(\cdot)\} \leq 0.25$  always, it follows that the sign of the ME equals the sign of  $\beta_j$  and that  $ME_j \leq 0.25 \times \beta_j$ .

The AMES following logit regression are

```
. * AMEs from logit regression
. margins, dydx(*)

Average marginal effects
Model VCE: Robust
Number of obs = 4,412

Expression: Pr(visit), predict()
dy/dx wrt: private chronic female income
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
private	.2258599	.0145665	15.51	0.000	.1973102	.2544097
chronic	.3251478	.0141019	23.06	0.000	.2975085	.352787
female	.1647049	.0122415	13.45	0.000	.1407119	.1886978
income	.0016927	.0002659	6.37	0.000	.0011716	.0022138

These AMES are very similar to those obtained after probit regression, differing only in the third significant digit. And from output not given, the predicted probabilities from the two models are very similar and have correlation 0.9998.

The logit model has the additional advantage that it implies that

$$\frac{\Pr(y = 1|\mathbf{x})}{\Pr(y = 0|\mathbf{x})} = \exp(\mathbf{x}'\boldsymbol{\beta})$$

The left-hand side is the odds ratio, the ratio of the event occurring to the event not occurring. A one-unit increase in  $x_j$  is associated with an approximate  $\exp(\beta_j)$  proportionate increase in the odds ratio. The `or` option of the `logit` command reports exponentiated coefficients.

## 10.6 Nonlinear least squares

Another common estimator of nonlinear models is the nonlinear least squares (NLS) estimator  $\hat{\beta}$ , which minimizes the sum of squared residuals

$$Q(\beta) = \sum_{i=1}^N \{y_i - m(\mathbf{x}_i, \beta)\}^2$$

where  $m(\mathbf{x}, \beta)$  is the specified functional form for  $E(y|\mathbf{x})$ , the conditional mean of  $y$  given  $\mathbf{x}$ .

If the conditional mean function is correctly specified, then the NLS estimator is consistent and asymptotically normally distributed. If the data-generating process is  $y_i = m(\mathbf{x}_i, \beta) + u_i$ , where the errors are independent and identically distributed  $(0, \sigma^2)$ , then NLS has desirable efficiency properties, and the NLS default estimate of the VCE is correct. Otherwise, a robust estimate of the VCE should be used.

### 10.6.1 The nl command

Command `nl` implements NLS regression. The simplest form of the command directly defines the conditional mean rather than calling a program or function and has syntax

```
nl (depvar=<sexp>) [if] [in] [weight] [, options]
```

where `<sexp>` is a substitutable expression. More details are provided in section [13.3.6](#) and in [R] `nl`. The only relevant option for our analysis here is option `vce()` for the type of estimate of the variance matrix of the estimates.

### 10.6.2 NLS for probit model

We consider application to the probit model, in which case  $m(\mathbf{x}_i, \beta) = \Phi(\mathbf{x}'_i \beta)$ . The `normal()` function computes the standard normal

c.d.f.  $\Phi(\cdot)$ , and the notation `xb:` is used to define a linear combination of variables; see section [13.3.6](#). As discussed in section [10.3.3](#), the `vce(robust)` option is used and we obtain

```
. * Nonlinear least-squares regression (command nl) for probit model
. nl (visit = normal({xb: private chronic female income}+{b0})), vce(robust)

Iteration 0: residual SS = 793.1719
Iteration 1: residual SS = 784.2946
Iteration 2: residual SS = 784.1461
Iteration 3: residual SS = 784.1458
Iteration 4: residual SS = 784.1458
Iteration 5: residual SS = 784.1458
Iteration 6: residual SS = 784.1458

Nonlinear regression                                         Number of obs =      4,412
                                                               R-squared =       0.7205
                                                               Adj R-squared =  0.7202
                                                               Root MSE =     .4218197
                                                               Res. dev. =    4899.035
```

visit	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
/xb_private	.7570137	.0536031	14.12	0.000	.6519248	.8621027
/xb_chronic	1.07188	.0564375	18.99	0.000	.9612344	1.182526
/xb_female	.5551205	.04543	12.22	0.000	.4660548	.6441861
/xb_income	.005946	.0009279	6.41	0.000	.0041269	.0077651
/b0	-.9598078	.0545463	-17.60	0.000	-1.066746	-.8528696

The `nl` coefficient estimates are similar to those from `probit` (within 2% for all regressors except `income`) given in section [10.3.2](#).

The `nl` robust standard errors are as much as 10% higher (for `chronic`). This efficiency loss is expected because, for a binary outcome, the NLS estimator differs from the fully efficient MLE that is obtained using the `probit` command.

At the same time, the NLS estimator is much better than the OLS estimator because it uses a model that will lead to predicted probabilities lying between zero and one.

The model diagnostic statistics given include  $R^2$  computed as the model (or explained) sum of squares divided by the total sum of squares, the root MSE that is the estimate  $s$  of the standard deviation  $\sigma$  of the model error, and

the deviance defined in section [13.8.3](#), which is a measure rarely used in econometrics. Note that  $R^2 = 0.7205$  is a very different measure than the probit model pseudo- $R^2 = 0.1940$ .

## 10.7 Other nonlinear estimators

In this section, we provide a brief summary of estimators detailed in the second volume.

When the density of the dependent variable is specified, estimation is by maximum likelihood because the MLE is fully efficient if the density is correctly specified. This is the case for the `probit` command, for example. Many of the chapters in volume 2 present commands that obtain ML estimates for the standard nonlinear regression models, such as multinomial models and count models.

When the specified density is not one already incorporated in Stata as a Stata command, one can still compute the MLE using the `mlexp` command, in the simplest cases, or the `ml` command. These commands require providing an algebraic expression for the log density. Chapters [13](#) and [16](#) provide details.

The generalized linear model (GLM) framework is the standard nonlinear model framework in many areas of applied statistics, most notably biostatistics. GLM estimators are essentially generalizations of least squares regression to nonlinear regression models for which there is a natural starting point for modeling the conditional mean and for modeling the intrinsic heteroskedasticity. GLMs cover many of the standard data types, including normal for  $y$  continuous on  $(-\infty, \infty)$ , gamma for  $y$  continuous on  $(0, \infty)$ , binomial and Bernoulli for number of successes in a given number of trials, and Poisson and negative binomial for count data with  $y = 0, 1, 2, \dots$ . GLM estimators have the important robustness-to-misspecification property that they are consistent provided only that the conditional mean function is correctly specified.

The `glm` command fits models in this class, where the particular model being used is specified as a command option. These models include the `probit` and `logit` models. The `glm` command provides more estimation and postestimation options than do model-specific commands such as the `probit` and `logit` commands. In particular, additional model diagnostics

can be obtained, and the `glm` command has an option to compute heteroskedastic autocorrelation consistent standard errors for time-series data. Section [13.3.8](#) provides further details.

Generalized methods of moments estimation for linear models was presented in chapters [7](#) and [9](#). The `gmm` command, which will be discussed in more detail in section [13.3.10](#), enables generalized methods of moments estimation in nonlinear models.

The nonlinear model used extensively in chapter [13](#) is one with exponential conditional mean, so  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ . Then the coefficients can be interpreted as semielasticities. For example, if  $\beta_j = 0.2$ , then a one-unit change in  $x_j$  is associated with a 0.2 proportionate change, or a 20% change, in  $E(y|\mathbf{x})$ . This model can be fit using the `poisson` command, even if  $y$  is not a count.

## 10.8 Additional resources

A complete listing of estimation commands can be obtained using `help estimation commands`. For a given estimation command such as `probit`, see the entries [R] **probit** and [R] **probit postestimation** and the corresponding online help.

Graduate econometrics texts give considerable detail on estimation and less on prediction and computation of MES.

Chapter [13](#) provides further detail on general methods for nonlinear models, including discussion of model diagnostics and the `n1`, `mlexp`, `m1`, `glm`, and `gmm` commands. Later chapters focus on the leading specific nonlinear models.

## 10.9 Exercises

1. You fit a probit model and estimate that  $\Pr(y = 1|x) = \Phi(-1 + 2x)$ . Give the formula for the ME of  $x$  on  $\Pr(y = 1|x)$  using calculus methods and using the finite-difference method. Then, compute the MEM using each method and the knowledge that  $\bar{x} = 1$ . Also, compute the MER at  $x = 2$  using each method. Comment on any differences. Given the above information, can the AME be computed? Explain.
2. Repeat the previous question for a logit model with  $\Pr(y = 1|x) = \Lambda(-1 + 2x)$ .
3. Perform probit regression similar to section [10.3.1](#) of visit on private, chronic, and income separately for the male and female samples, using factor variables for the discrete regressors. Compare coefficient estimates across the two samples. Compare the AME across the two samples. Which, if any, is the more meaningful comparison? Perform a test at a significance level of 0.05 of whether there is a difference across the two samples. Hint: Nest the two samples in a larger model.
4. Run the full-sample logit and probit regressions of visit on private, chronic, female, and income. Use the predict command to generate fitted probabilities for each regression. Use the twoway scatter graphing command to plot a graph of the logit predictions on the probit predictions. What do you find? Comment.
5. In this question, use 1997 data, rather than 2002 data. From section [10.4.7](#), the linear probability model has deficiencies. Nonetheless, fit the model by linear regression of visit on private, chronic, female, and income. Also, fit the corresponding logit model, using factor variables. Compare the AMEs across the two models and comment. Next, compare the MEMs across the two models and with the preceding AMEs. Comment.
6. After fitting the linear probability and logit models of the preceding question, generate the predicted values using the predict postestimation command. Show that the average difference between the logit predictions and linear probability predictions is close to zero and that their correlation with each other is quite high. Check whether any of the fitted probabilities are outside the (0, 1) interval. Comment.

Use the `twoway scatter` graphing command to plot a graph of the OLS predictions on the logit predictions. What do you find? Comment.

7. Using as a template the NLS approach to fitting the probit model presented in section [10.6.2](#), fit a logit model. This requires substituting the c.d.f. `logistic` in place of the c.d.f. `normal`. Compare the estimated coefficients with those from the `logit` command. Would you expect the AME from this NLS regression to differ significantly from those for probit regression? Why or why not?
8. Suppose you are given a bivariate sample  $(y_i, x_i)$  of two positive-valued observations. Two functional forms are suggested for a regression model: 1)  $\ln y = \beta_0 + \beta_1 \ln x + e_1$ , and 2) a nonlinear regression  $y = \alpha_0 x^{\alpha_1} + e_2$ . Suppose that neither is preferred a priori and goodness-of-fit criterion carries high weight in the final selection. Assume that the errors are draws from normal zero-mean homoskedastic distributions. Use suitable values of the parameters to generate two samples, based on model 1 and model 2, respectively. Each model is treated as an approximation when the other model has generated the data. Use OLS to fit model (i) and NLS to fit model (ii). Evaluate the performance of each misspecified (“wrong”) model in terms of goodness-of-fit or within-sample prediction.



# **Chapter 11**

## **Tests of hypotheses and model specification**

## 11.1 Introduction

Econometric modeling is composed of a cycle of initial model specification, estimation, diagnostic checks, and model respecification. The diagnostic checks are often based on hypothesis tests for the statistical significance of key variables and model-specification tests. This chapter presents additional details on hypothesis tests, associated confidence intervals, and model-specification tests that are used widely throughout the book.

The emphasis is on Wald hypothesis tests and confidence intervals, the most commonly used inference methods in microeconomics. These produce the standard regression output and can also be obtained by using the `test`, `testnl`, `lincom`, and `nlcom` commands. We also present the other two classical testing methods, likelihood-ratio (LR) and Lagrange multiplier (LM) tests.

We then present familywise error rates (FWER) and false discovery rates (FDRS) when multiple tests are performed, such as testing for statistical significance in each of several subgroups or for the impact of a key regressor on each of several outcomes. Failure to adjust test size in such cases leads to great understatement of true test size and hence false discoveries of statistical significance.

In discussing results in this book, we have in many places stated whether a regressor or effect is statistically significant at 5%. One should realize that we have done this for brevity and convenience. In a real empirical study, one should instead emphasize the actual size of the effect (the “economic” significance) with a measure of uncertainty such as a standard error or confidence interval. For discussion of the weaknesses of considering only statistical significance and  $p$ -values, see [Wasserstein and Lazar \(2016\)](#) and the many articles in [Wasserstein, Schirm, and Lazar \(2019\)](#).

When statistical significance is reported, there is growing concern among statisticians that  $p$ -values reported in published studies may underestimate the true  $p$ -value. In practice, considerable pretesting may occur

before final model specification, and standard inference methods fail to consider the effect of this pretesting on size and power of any ultimate test. While the effects of such data mining can be minimized by relying on economic theory and previous studies in determining model specification, or by using a different sample from the final estimation sample, this is not always done. Furthermore, *p*-hacking and publication bias can lead to disproportionately favoring statistically significant results. For example, meta-analyses find a bunching of *p*-values from published studies in the 0.04–0.05 range. [Brodeur, Cook, and Heyes \(2020\)](#) consider causal studies in leading economics journals and find this bunching is more prevalent in instrumental variables (iv) and difference-in-differences studies than in randomized control trials and regression discontinuity design studies and in all these cases is less prevalent than in some other social sciences. Methods to compute *p*-values that control for multiple testing are presented in section [11.6](#).

Economic studies typically report test size with no consideration of test power. More recently, attention has turned to test power. One reason is that laboratory and field experiments have become more common, and these experiments need to be designed to have reasonable power, the standard threshold being that a test of size 0.05 should have power of at least 0.8 against an alternative hypothesis of a desired effect size.

We give considerable discussion of test size and power. Monte Carlo methods for obtaining test size and power are presented. The `power.onemean` command can be adapted to regression settings to calculate test power for given effect size, minimum effect size for desired power, and minimum sample size for desired power.

The chapter then presents a brief general discussion of model-specification tests, including information matrix (IM) tests, goodness-of-fit tests, Hausman tests, and tests of overidentifying restrictions, that are applied in various chapters. Model-selection tests for nonnested nonlinear models are not covered, though some of the methods given in chapter [3](#) for linear models can be extended to nonlinear models, and a brief discussion for likelihood-based nonlinear models is given in section [13.8.2](#). The chapter concludes with a discussion of permutation and randomization tests.

This chapter uses Poisson regression, which will be discussed in more detail in section [13.2](#), as the leading example, so that we cover both linear and nonlinear models. In most places, the methods for ordinary least squares (OLS) are the same; simply change `poisson` to `regress`.

## 11.2 Critical values and p-values

Before discussing Stata estimation and testing commands and associated output, we discuss how critical values and  $p$ -values are computed.

Introductory econometrics courses often emphasize use of the  $t(n)$  and  $F(h, n)$  distributions for hypothesis testing, where  $n$  is the degrees of freedom and  $h$  is the number of restrictions. For cross-sectional analysis, often  $n = N - K$ , where  $N$  is the sample size and  $K$  is the number of regressors. For clustered data, Stata sets  $n = G - 1$ , where  $G$  is the number of clusters.

These distributions hold exactly only in the very special case of tests of linear restrictions for the OLS estimator in the linear regression model with independent normal homoskedastic errors. Instead, virtually all inference in microeconomics is based on asymptotic theory. This is the case not only for nonlinear estimators but also for linear estimators, such as OLS and IV, with robust standard errors. Then test statistics are asymptotically standard normal,  $Z$ , distributed rather than  $t(n)$  and chi-squared,  $\chi^2(h)$ , distributed rather than  $F(h, n)$ .

### 11.2.1 Standard normal compared with Student's t

The change from  $t(n)$  to standard normal distributions is relatively minor, unless  $n$  is small, say, less than 30. The two distributions are identical for  $n \rightarrow \infty$ . The  $t$  distribution has fatter tails, leading to larger  $p$ -values and critical values than the standard normal at conventional levels of significance such as 0.05.

For clustered data with  $G$  clusters, studies find that it is much better to use the  $t(G - 1)$  distribution rather than the standard normal. When there are few clusters, a not unusual occurrence, the consequent  $p$ -values and critical values can differ substantially.

### 11.2.2 Chi-squared compared with F

Many tests of joint hypotheses use the  $\chi^2$  distribution. A  $\chi^2(h)$  random variable has a mean of  $h$  and a variance of  $2h$ , and for  $h > 7$ , the 5% critical value lies between  $h$  and  $2h$ .

The  $\chi^2(h)$  distribution is scaled quite differently from the  $F$ . As the denominator degrees of freedom of the  $F$  goes to infinity, we have

$$F(h, n) \rightarrow \frac{\chi^2(h)}{h} \text{ as } n \rightarrow \infty \quad (11.1)$$

Thus, if asymptotic theory leads to a test statistic that is  $\chi^2(h)$  distributed, then division of this statistic by  $h$  leads to a statistic that is approximately  $F(h, n)$  distributed if  $n$  is large. In finite samples, the  $F(h, n)$  distribution has fatter tails than  $\chi^2(h)/h$ , leading to larger  $p$ -values and critical values for the  $F$  compared with the  $\chi^2$ .

### 11.2.3 Plotting densities

We compare the density of a  $\chi^2(5)$  random variable with a random variable that is 5 times an  $F(5, 30)$  random variable. From (11.1), the two are the same for large  $n$  but will differ for  $n = 30$ . In practice,  $n = 30$  is not large enough for asymptotic theory to approximate the finite distribution well.

One way to compare is to evaluate the formulas for the respective densities at a range of points, say, 0.1, 0.2, ..., 20.0, and graph the density values against the evaluation points. The `graph twoway function` command automates this method. This is left as an exercise.

This approach requires providing density formulas that can be quite complicated and may even be unknown to the user if the density is that of a mixture distribution, for example. A simpler way is to make many draws from the respective distributions, using the methods of section 5.2, and then use the `kdensity` command to compute and graph the kernel density estimate.

We take this latter approach. We begin by taking 10,000 draws from each distribution. We use the `rchi2()` function; see section [5.2](#).

```
. * Create many draws from chi(5) and 5*F(5,30) distributions
. set seed 10101
. qui set obs 10000
. generate chi5 = rchi2(5)                      // Result xc ~ chisquared(10)
. generate xfn = rchi2(5)/5                     // Result numerator of F(5,30)
. generate xfd = rchi2(30)/30                  // Result denominator of F(5,30)
. generate f5_30 = xfn/xfd                     // Result xf ~ F(5,30)
. generate five_x_f5_30 = 5*f5_30
. summarize chi5 five_x_f5_30
```

Variable	Obs	Mean	Std. dev.	Min	Max
chi5	10,000	4.994329	3.169312	.103002	33.19207
five_x_f5_30	10,000	5.322791	3.790381	.0525943	31.24482

For `chi5`, the average of 4.99 is close to the theoretical mean of 5, and the sample variance of  $3.1693^2 = 10.04$  is close to the theoretical variance of 10. For `five_x_f5_30`, the sample variance of  $3.790^2 = 14.36$  is much larger than that of `chi5`, reflecting the previously mentioned fatter tails.

We then plot the kernel density estimates based on these draws. To improve graph readability, we plot the kernel density estimates only for draws less than 25, using a method already explained in section [3.2.8](#). To produce smoother plots, we increase the default bandwidth to 1.0, an alternative being to increase the number of draws. We have

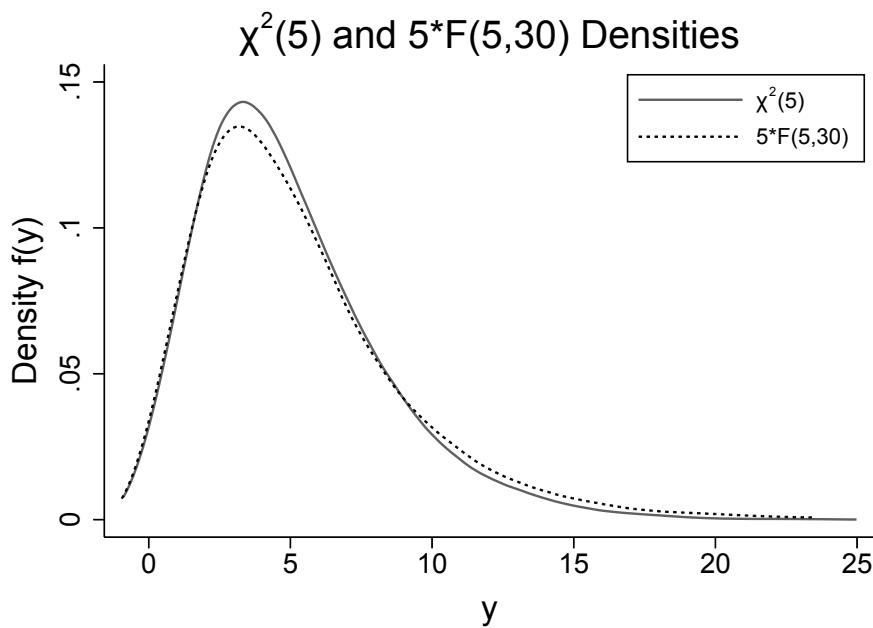
```
. * Plot the densities for these two distributions using kdensity
. label var chi5 "chi(5)"
. label var five_x_f5_30 "5*F(5,30)"
. kdensity chi5, bw(1.0) generate(kx1 kd1) n(500)
. kdensity five_x_f5_30, bw(1.0) generate(kx2 kd2) n(500)
. qui drop if (chi5 > 25 | five_x_f5_30 > 25)
```

```

. graph twoway (line kd1 kx1) (line kd2 kx2, clstyle(p3)) if kx1 < 25,
>     scale(1.2) plotregion(style(none))
>     title("{$\chi^2$}(5) and 5*F(5,30) Densities")
>     xtitle("y", size(medlarge)) xscale(titlegap(*5))
>     ytitle("Density f(y)", size(medlarge)) yscale(titlegap(*5))
>     legend(pos(1) ring(0) col(1)) legend(size(small))
>     legend(label(1 "{$\chi^2$}(5)") label(2 "5*F(5,30)"))

```

In figure 11.1, the two densities appear similar, though the density of  $5 \times F(5, 30)$  has a longer tail than that of  $\chi^2(5)$ , and it is the tails that are used for tests at a 0.05 level and for 95% confidence intervals. The difference disappears as the denominator degrees of freedom (here 30) goes to infinity.



**Figure 11.1.**  $\chi^2(5)$  density compared with 5 times  $F(5, 30)$  density

#### 11.2.4 Computing p-values and critical values

Stata output automatically provides  $p$ -values but not critical values. The  $p$ -values can be obtained manually from the relevant cumulative distribution function (c.d.f.), whereas critical values can be obtained by using the inverse c.d.f. The precise Stata functions vary with the distribution. For details, see [FN] **Statistical functions** or type `help density functions`.

We compute  $p$ -values for the test of a single restriction ( $h = 1$ ). We suppose the test statistic is equal to 2 and use the  $t(30)$  or  $Z$  distributions. In that case, it is equivalently equal to  $2^2 = 4$ , and we use the  $F(1, 30)$  or  $\chi^2(1)$  distribution. We have

```
. * p-values for t(30), F(1,30), Z, and chi(1) at y = 2
. scalar y = 2
. scalar p_t30 = 2*ttail(30,y)
. scalar p_f1and30 = Ftail(1,30,y^2)
. scalar p_z = 2*(1-normal(y))
. scalar p_chi1 = chi2tail(1,y^2)
. display "p-values" " t(30) =" %7.4f p_t30 " F(1,30)=" %7.4f
>     p_f1and30 " z =" %7.4f p_z " chi(1)=" %7.4f p_chi1
p-values t(30) = 0.0546 F(1,30)= 0.0546 z = 0.0455 chi(1)= 0.0455
```

The general properties that  $Z^2 = \chi^2(1)$  and  $t(n)^2 = F(1, n)$  are confirmed for this example. Also,  $t(n) \rightarrow Z$  and  $F(1, n)/1 \rightarrow \chi^2(1)$  as  $n \rightarrow \infty$ , but there is still a difference for  $n = 30$ , with a  $p$ -value of 0.0455 compared with 0.0546.

We next compute critical values for these distributions for a two-sided test of a single restriction at a level of 0.05. We have

```
. * Critical values for t(30), F(1,30), Z, and chi(1) at level 0.05
. scalar alpha = 0.05
. scalar c_t30 = invttail(30,alpha/2)
. scalar c_f1and30 = invFtail(1,30,alpha)
. scalar c_z = -invnormal(alpha/2)
. scalar c_chi1 = invchi2(1,1-alpha)
. display "critical values" " t(30) =" %7.3f c_t30 " F(1,30)=" %7.3f
>     c_f1and30 " z =" %7.3f c_z " chi(1)=" %7.3f c_chi1
critical values t(30) = 2.042 F(1,30)= 4.171 z = 1.960 chi(1)= 3.841
```

Again,  $t(30)^2 = F(1, 30)$  and  $Z^2 = \chi^2(1)$ , whereas  $t(30) \simeq Z$  and  $F(1, 30)/1 \simeq \chi^2(1)$ .

## 11.2.5 Which distributions does Stata use?

In practice, the  $t$  and  $F$  distributions may continue to be used as an ad hoc finite-sample correction, even when only asymptotic results supporting the  $Z$  and  $\chi^2$  distributions are available. This leads to more conservative inference, with less likelihood of rejecting the null hypothesis because  $p$ -values are larger and with wider confidence intervals because critical values are larger.

For estimators of linear regression models, Stata uses the  $t(N - K)$  and  $F(q, N - K)$  distributions for independent observations and, in some cases, the  $t(G - 1)$  and  $F(q, G - 1)$  distributions for clustered data with  $G$  clusters. For estimators of nonlinear regression models, Stata uses the  $z$  and  $\chi^2(q)$  distributions for both independent observations and clustered observations.

For clustered data, it is better to use the  $t(G - 1)$  and  $F(q, G - 1)$  distributions. Studies find that even this leads to tests that overreject, though less so than if the  $z$  and  $\chi^2(q)$  distributions are used.  $F$  tests can always be implemented using the `df(#)` option of the `test` and `testparm` postestimation commands.

## 11.3 Wald tests and confidence intervals

A quite universal method for hypothesis testing and obtaining confidence intervals is the Wald method, based on the estimated variance–covariance matrix of the estimator (VCE) presented in sections [3.4](#) and [13.4](#). This method produces the test statistics and  $p$ -values for a test of the significance of individual coefficients, the confidence intervals for individual coefficients, and the tests of overall significance that are given in Stata regression output.

Here we provide background on the Wald test, extension to tests of more complicated hypotheses that require the use of the `test` and `testnl` commands, and extension to confidence intervals on combinations of parameters using the `lincom` and `nlcom` commands.

### 11.3.1 Wald test of linear hypotheses

By a linear hypothesis, we mean one that can be expressed as a linear combination of parameters. Single hypothesis examples include  $H_0: \beta_2 = 0$  and  $H_0: \beta_2 - \beta_3 - 5 = 0$ . A joint hypothesis example tests the two preceding hypotheses simultaneously.

The Wald test method is intuitively appealing. The test is based on how well the corresponding parameter estimates satisfy the null hypothesis. For example, to test  $H_0: \beta_2 - \beta_3 - 5 = 0$ , we ask whether  $\widehat{\beta}_2 - \widehat{\beta}_3 - 5 \simeq 0$ . To implement the test, we need to know the distribution of  $\widehat{\beta}_2 - \widehat{\beta}_3 - 5$ . But this is easy because the estimators used in this book are usually asymptotically normal, and a linear combination of normals is normal.

We do need to find the variance of this normal distribution. In this example,  $\text{Var}(\widehat{\beta}_2 - \widehat{\beta}_3 - 5) = \text{Var}(\widehat{\beta}_2) + \text{Var}(\widehat{\beta}_3) - 2\text{Cov}(\widehat{\beta}_2, \widehat{\beta}_3)$  because for the random variables  $X$  and  $Y$ ,  $\text{Var}(X - Y) = \text{Var}(X) + \text{Var}(Y) - 2\text{Cov}(X, Y)$ . More generally, it is helpful to use matrix notation, which we now introduce.

Let  $\boldsymbol{\beta}$  denote the  $K \times 1$  parameter vector, where the results also apply if instead we use the more general notation  $\boldsymbol{\theta}$ , which includes  $\boldsymbol{\beta}$  and any auxiliary parameters. Then, for example,  $H_0: \beta_2 = 0$  and  $\beta_2 - \beta_3 - 5 = 0$  can be written as

$$\begin{bmatrix} \beta_2 \\ \beta_2 - \beta_3 - 5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_k \end{bmatrix} - \begin{bmatrix} 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This linear combination can be written as  $\mathbf{R}\boldsymbol{\beta} - \mathbf{r} = \mathbf{0}$ .

For a two-sided test of  $h$  linear hypotheses under  $H_0$ , we therefore test

$$\begin{aligned} H_0: \mathbf{R}\boldsymbol{\beta} - \mathbf{r} &= \mathbf{0} \\ H_a: \mathbf{R}\boldsymbol{\beta} - \mathbf{r} &\neq \mathbf{0} \end{aligned}$$

where  $\mathbf{R}$  is an  $h \times K$  matrix and  $\mathbf{r}$  is an  $h \times 1$  vector,  $h \leq K$ . Standard examples include tests of individual exclusions restrictions,  $\beta_j = 0$ , and tests of joint statistical significance,  $\beta_2 = 0, \dots, \beta_q = 0$  (with  $\beta_1$  as an intercept coefficient).

The Wald test uses the quite intuitive approach of rejecting  $H_0: \mathbf{R}\boldsymbol{\beta} - \mathbf{r} = \mathbf{0}$  if  $\mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r}$  is considerably different from  $\mathbf{0}$ . Now,

$$\begin{aligned} \widehat{\boldsymbol{\beta}} &\stackrel{a}{\sim} N\left\{\boldsymbol{\beta}, \text{Var}(\widehat{\boldsymbol{\beta}})\right\} \\ \implies \mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r} &\stackrel{a}{\sim} N\left\{\mathbf{R}\boldsymbol{\beta} - \mathbf{r}, \mathbf{R}\text{Var}(\widehat{\boldsymbol{\beta}})\mathbf{R}'\right\} \\ \implies \mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r} &\stackrel{a}{\sim} N\left\{\mathbf{0}, \mathbf{R}\text{Var}(\widehat{\boldsymbol{\beta}})\mathbf{R}'\right\} \quad \text{under } H_0 \end{aligned} \tag{11.2}$$

For a single hypothesis,  $\mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r}$  is a scalar that is univariate normally distributed, so we can transform to a standard normal variate and use standard normal tables.

More generally, there are multiple hypotheses. To avoid using the multivariate normal distribution, we transform to a chi-squared distribution. If the  $h \times 1$  vector  $\mathbf{y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , then  $(\mathbf{y} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu}) \sim \chi^2(h)$ . Applying this result to (11.2), we obtain the Wald statistic for the test of  $H_0: \mathbf{R}\boldsymbol{\beta} - \mathbf{r} = \mathbf{0}$ :

$$W = (\mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r})' \left\{ \mathbf{R}\widehat{\boldsymbol{V}}(\widehat{\boldsymbol{\beta}}) \mathbf{R}' \right\}^{-1} (\mathbf{R}\widehat{\boldsymbol{\beta}} - \mathbf{r}) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0 \tag{11.3}$$

Large values of  $W$  lead to rejection of  $H_0$ . At a level of 0.05, for example, we reject  $H_0$  if the  $p$ -value  $p = \Pr\{\chi^2(h) > W\} < 0.05$  or if  $W$  exceeds the critical value  $c = \chi^2_{0.05}(h)$ , where by  $\chi^2_{0.05}(h)$  we mean the area in the right tail is 0.05.

In going from (11.2) to (11.3), we also replaced  $\text{Var}(\hat{\beta})$  by an estimate,  $\hat{V}(\hat{\beta})$ . For the test to be valid, the estimate  $\hat{V}(\hat{\beta})$  must be consistent for  $\text{Var}(\hat{\beta})$ ; that is, we need to use a correct estimator of the VCE.

An alternative test statistic is the  $F$  statistic, which is the Wald statistic divided by the number of restrictions. Then,

$$F = \frac{W}{h} \stackrel{a}{\sim} F(h, N - K) \text{ under } H_0 \quad (11.4)$$

where  $K$  denotes the number of parameters in the regression model. Large values of  $F$  lead to rejection of  $H_0$ . At a level of 0.05, for example, we reject  $H_0$  if the  $p$ -value  $p = \Pr\{F(h, N - K) > F\} < 0.05$  or if  $F$  exceeds the critical value  $c = F_{0.05}(h, N - K)$ .

### 11.3.2 The `test` and `testparm` commands

The Wald test can be performed by using the `test` command or the `testparm` command.

The `test` command has several different syntaxes. The simplest two are

```
test coeflist
test exp = exp [= ...]
```

The syntax is best explained with examples. More complicated syntax enables testing across equations in multiequation models. A multiequation example following the `sureg` command was given in section 6.8. An example following `nbreg` is given as an end-of-chapter exercise.

In simple cases, the `coeflist` can be a list of regressor names, but in more complex cases, it needs to be a list of coefficient names. It can be difficult to know the Stata convention for naming coefficients in some cases. Using the estimation command option `coeflegend` or using the output from the postestimation command `estat vce` may give the appropriate complete names.

In such situations, it may be easier to use the `testparm` command, which requires a list of variable names, rather than coefficient names, and has syntax

```
testparm varlist [ , options ]
```

The `testparm` command is especially useful when the estimation command uses factor-variable notation.

Usually, the  $W$  statistic in (11.3) based on the chi-squared distribution, is used, though the  $F$  statistic in (11.4) is used after fitting linear models. However, when cluster-robust standard errors are used and there are few clusters, using the chi-squared distribution, or  $N(0, 1)$  for a single hypothesis, leads to substantial overrejection; see section 6.4.6. At a minimum, one should then use the `test` command or `testparm` with the `df(#)` option, where  $\#$  equals  $G - 1$ , where  $G$  is the number of clusters. This implements an  $F$  statistic with  $F(h, G - 1)$  degrees of freedom, where  $h$  is the number of hypotheses. The wild cluster bootstrap for few clusters is presented in section 12.6.

The other options of the `testparm` command are usually not needed. They include `mtest` to test each hypothesis separately if several hypotheses are given and `accumulate` to test hypotheses jointly with previously tested hypotheses.

### 11.3.3 Data example

We illustrate the Wald test, and subsequent tests, using the dataset from the 2002 U.S. Medical Expenditure Panel Survey first used in chapter 10. We model the number of office-based physician visits (`docvis`) by persons aged 25–64 years. The regressors are restricted to health insurance status (`private`), health status (`chronic`), and socioeconomic characteristics (`female` and `income`) to keep Stata output short.

We consider a nonlinear estimator, the Poisson quasi-MLE, which will be discussed in more detail in section 13.2.2. The coefficients are interpreted as semielasticities; see section 13.3.2. We have

```

. * Fit Poisson model used throughout this chapter
. qui use mus210mepsdocvisyoung, clear
. qui keep if year02==1
. poisson docvis private chronic female income, vce(robust) nolog
Poisson regression
Number of obs = 4,412
Wald chi2(4) = 594.72
Prob > chi2 = 0.0000
Pseudo R2 = 0.1930
Log pseudolikelihood = -18503.549

```

docvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	.7986652	.1090014	7.33	0.000	.5850263	1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167	1.201614
female	.4925481	.0585365	8.41	0.000	.3778187	.6072774
income	.003557	.0010825	3.29	0.001	.0014354	.0056787
_cons	-.2297262	.1108732	-2.07	0.038	-.4470338	-.0124186

### Test single coefficient

To test whether a single coefficient equals zero, we just need to specify the regressor name. For example, to test  $H_0: \beta_{\text{female}} = 0$ , we have

```

. * Test a single coefficient equal 0
. test female
( 1) [docvis]female = 0
chi2( 1) = 70.80
Prob > chi2 = 0.0000

```

We reject  $H_0$  because  $p < 0.05$  and conclude that `female` is statistically significant at the level of 0.05. The test statistic is the square of the  $z$  statistic given in the regression output ( $8.414^2 = 70.80$ ), and the  $p$ -values are the same.

### Test several hypotheses

As an example of testing more than one hypothesis, we test  $H_0: \beta_{\text{female}} = 0$  and  $\beta_{\text{private}} + \beta_{\text{chronic}} = 1$ . Then,

```

. * Test two hypotheses jointly using test
. test (female) (private + chronic = 1)
( 1) [docvis]female = 0
( 2) [docvis]private + [docvis]chronic = 1
chi2( 2) = 122.29
Prob > chi2 = 0.0000

```

We reject  $H_0$  because  $p < 0.05$ .

The `mtest` option additionally tests each hypothesis in isolation. We have

```
. * Test each hypothesis in isolation as well as jointly
. test (female) (private + chronic = 1), mtest
( 1) [docvis]female = 0
( 2) [docvis]private + [docvis]chronic = 1
```

	chi2	df	p > chi2
(1)	70.80	1	0.0000*
(2)	56.53	1	0.0000*
All	122.29	2	0.0000

\* Unadjusted *p*-values

As expected, the hypothesis test value of 70.80 for `female` equals that given earlier when the hypothesis was tested in isolation.

The preceding test makes no adjustment to *p*-values to account for multiple testing. Options to `mtest` include several that implement Bonferroni's method and variations. This extension is detailed in section [11.6](#).

### Test of overall significance

The `test` command can be used to test overall significance. We have

```
. * Wald test of overall significance
. test private chronic female income
( 1) [docvis]private = 0
( 2) [docvis]chronic = 0
( 3) [docvis]female = 0
( 4) [docvis]income = 0
      chi2( 4) =  594.72
      Prob > chi2 =    0.0000
```

The Wald test statistic value of 594.72 is the same as that given in the `poisson` output.

### Test calculated from retrieved coefficients and VCE

For pedagogical purposes, we compute this overall test manually even though we use `test` in practice. The computation requires retrieving  $\hat{\beta}$  and  $\hat{V}(\hat{\beta})$ , defining the appropriate matrices  $\mathbf{R}$  and  $\mathbf{r}$ , and calculating  $W$  defined in [\(11.3\)](#). In doing so, we note that Stata stores regression coefficients as a row vector, so we need to transpose to get the  $K \times 1$  column vector  $\hat{\beta}$ . Because we use Stata estimates of  $\hat{\beta}$

and  $\widehat{V}(\widehat{\beta})$ , in defining  $\mathbf{R}$  and  $\mathbf{r}$ , we need to follow the Stata convention of placing the intercept coefficient as the last coefficient. We have

```
. * Manually compute overall test of significance using the formula for W
. qui poisson docvis private chronic female income, vce(robust)
. matrix b = e(b)'
. matrix V = e(V)
. matrix R = (1,0,0,0,0 \ 0,1,0,0,0 \ 0,0,1,0,0 \ 0,0,0,1,0 )
. matrix r = (0 \ 0 \ 0 \ 0)
. matrix W = (R*b-r)'*invsym(R*V*R')*(R*b-r)
. scalar Wald = W[1,1]
. scalar h = rowsof(R)
. display "Wald test statistic: " Wald " with p-value: " chi2tail(h,Wald)
Wald test statistic: 594.72457 with p-value: 2.15e-127
```

The value of 594.72 is the same as that from the `test` command.

### 11.3.4 One-sided Wald tests

The preceding tests are two-sided tests, such as  $\beta_j = 0$  against  $\beta_j \neq 0$ . We now consider one-sided tests of a single hypothesis, such as a test of whether  $\beta_j > 0$ .

The first step in conducting a one-sided test is determining which side is  $H_0$  and which side is  $H_a$ . The convention is that the claim made is set as the alternative hypothesis. For example, if the claim is made that the  $j$ th regressor has a positive marginal effect and this means that  $\beta_j > 0$ , then we test  $H_0: \beta_j \leq 0$  against  $H_a: \beta_j > 0$ .

The second step is to obtain a test statistic. For tests on a single regressor, we use the  $z$  statistic

$$z = \frac{\widehat{\beta}_j}{s_{\widehat{\beta}_j}} \stackrel{a}{\sim} N(0, 1) \text{ under } H_0$$

where  $z^2 = W$  given in (11.3). In some cases, the  $t(N - K)$  distribution is used, in which case the  $z$  statistic is called a  $t$  statistic. Regression output gives this statistic, along with  $p$ -values for two-sided tests. For a one-sided test, these  $p$ -values should be halved, with the important condition that it is necessary to check that  $\widehat{\beta}_j$  has the correct sign. For example, if testing  $H_0: \beta_j \leq 0$  against  $H_a: \beta_j > 0$ , then we reject

$H_0$  at the level of 0.05 if  $\widehat{\beta}_j > 0$ , and the reported two-sided  $p$ -value is less than 0.10. If instead  $\widehat{\beta}_j < 0$ , the  $p$ -value for a one-sided test must be at least 0.50 because we are on the wrong side of 0, leading to certain inability to reject at conventional statistical significance levels.

As an example, consider a test of the claim that doctor visits increase with income, even after controlling for chronic conditions, gender, and income. The appropriate test of this claim is one of  $H_0: \beta_{\text{income}} \leq 0$  against  $H_a: \beta_{\text{income}} > 0$ . The `poisson` output includes  $\widehat{\beta}_{\text{income}} = 0.0036$  with  $p = 0.001$  for a two-sided test. Because  $\widehat{\beta}_{\text{income}} > 0$ , we simply halve the two-sided test  $p$ -value to get  $p = 0.001/2 = 0.0005 < 0.05$ . So we reject  $H_0: \beta_{\text{income}} \leq 0$  at the 0.05 level.

More generally, suppose we want to test the single hypothesis  $H_0: \mathbf{R}\boldsymbol{\beta} - r \leq 0$  against  $H_a: \mathbf{R}\boldsymbol{\beta} - r > 0$ , where here  $\mathbf{R}\boldsymbol{\beta} - r$  is a scalar. Then we use

$$z = \frac{\mathbf{R}\widehat{\boldsymbol{\beta}} - r}{s_{\mathbf{R}\widehat{\boldsymbol{\beta}} - r}} \stackrel{a}{\sim} N(0, 1) \text{ under } H_0$$

When squared, this statistic again equals the corresponding Wald test; that is,  $z^2 = W$ . The `test` command gives  $W$ , but  $z$  could be either  $\sqrt{W}$  or  $-\sqrt{W}$ , and the sign of  $z$  is needed to perform the one-sided test. To obtain the sign, we can also compute  $\mathbf{R}\widehat{\boldsymbol{\beta}} - r$  by using the `lincom` command; see section 11.3.10. If  $\mathbf{R}\widehat{\boldsymbol{\beta}} - r$  has a sign that differs from that of  $\mathbf{R}\boldsymbol{\beta} - r$  under  $H_0$ , then the  $p$ -value is one half of the two-sided  $p$ -value given by `test` (or by `lincom`); we reject  $H_0$  at the  $\alpha$  level if this adjusted  $p$ -value is less than  $\alpha$  and do not reject otherwise. If instead  $\mathbf{R}\widehat{\boldsymbol{\beta}} - r$  has the same sign as that of  $\mathbf{R}\boldsymbol{\beta} - r$  under  $H_0$ , then we always do not reject  $H_0$ .

### 11.3.5 Wald test of nonlinear hypotheses (delta method)

Not all hypotheses are linear combinations of parameters. A nonlinear hypothesis example is a test of  $H_0: \beta_2/\beta_3 = 1$  against  $H_a: \beta_2/\beta_3 \neq 1$ . This can be expressed as a test of  $g(\boldsymbol{\beta}) = 0$ , where  $g(\boldsymbol{\beta}) = \beta_2/\beta_3 - 1$ . More generally, there can be  $h$  hypotheses combined into the  $h \times 1$  vector  $\mathbf{g}(\boldsymbol{\beta}) = \mathbf{0}$ , with each separate hypothesis being a separate row in  $\mathbf{g}(\boldsymbol{\beta})$ . Linear hypotheses are the special case of  $\mathbf{g}(\boldsymbol{\beta}) = \mathbf{R}\boldsymbol{\beta} - \mathbf{r}$ .

The Wald test method is now based on the closeness of  $\mathbf{g}(\widehat{\boldsymbol{\beta}})$  to  $\mathbf{0}$ . Because  $\widehat{\boldsymbol{\beta}}$  is asymptotically normal, so too is  $\mathbf{g}(\widehat{\boldsymbol{\beta}})$ . Some algebra that includes linearization of

$\mathbf{g}(\hat{\boldsymbol{\beta}})$  using a Taylor-series expansion yields the Wald test statistic for the nonlinear hypotheses  $H_0: \mathbf{g}(\boldsymbol{\beta}) = \mathbf{0}$ :

$$W = \mathbf{g}(\hat{\boldsymbol{\beta}})' \left\{ \widehat{\mathbf{R}} \widehat{V}(\hat{\boldsymbol{\beta}}) \widehat{\mathbf{R}}' \right\}^{-1} \mathbf{g}(\hat{\boldsymbol{\beta}}) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0, \text{ where } \widehat{\mathbf{R}} = \frac{\partial \mathbf{g}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'} \Big|_{\hat{\boldsymbol{\beta}}} \quad (11.5)$$

This is the same test statistic as  $W$  in (11.3) upon replacement of  $\mathbf{R}\hat{\boldsymbol{\beta}} - r$  by  $\mathbf{g}(\hat{\boldsymbol{\beta}})$  and replacement of  $\mathbf{R}$  by  $\widehat{\mathbf{R}}$ . Again, large values of  $W$  lead to rejection of  $H_0$ , and  $p = \Pr\{\chi^2(h) > W\}$ .

The test statistic is often called one based on the delta method because of the derivative used to form  $\widehat{\mathbf{R}}$ .

### 11.3.6 The `testnl` command

The Wald test for nonlinear hypotheses is performed using the `testnl` command. The basic syntax is

```
testnl exp = exp [= exp ...] [ , options ]
```

The main option is `mtest` to separately test each hypothesis in a joint test.

As an example, we consider a test of  $H_0: \beta_{\text{female}}/\beta_{\text{private}} - 1 = 0$  against  $H_a: \beta_{\text{female}}/\beta_{\text{private}} - 1 \neq 0$ . Then,

```
. * Test a nonlinear hypothesis using testnl
. testnl _b[female]/_b[private] = 1
(1) _b[female]/_b[private] = 1
               chi2(1) =      13.51
               Prob > chi2 =    0.0002
```

We reject  $H_0$  at the 0.05 level because  $p < 0.05$ .

The hypothesis in the preceding example can be equivalently expressed as  $\beta_{\text{female}} = \beta_{\text{private}}$ . So a simpler test is

```
. * Wald test is not invariant
. test female = private
( 1) - [docvis]private + [docvis]female = 0
               chi2( 1) =     6.85
               Prob > chi2 =   0.0088
```

Surprisingly, we get different values for the test statistic and  $p$ -value, even though both methods are valid and are asymptotically equivalent. This illustrates a weakness of Wald tests: in finite samples, they are not invariant to nonlinear transformations of the null hypothesis. With one representation of the null hypothesis, we might reject  $H_0$  at the  $\alpha$  level, whereas with a different representation we might not. LR and LM tests do not have this weakness.

### 11.3.7 Forward and backward selection based on statistical significance

Forward selection, a specific-to-general approach, starts with the simplest model, an intercept-only model, and sequentially adds the most highly statistically significant regressor, provided this regressor is statistically significant at the prespecified significance level.

The `stepwise` prefix with the `pe()` option implements forward selection.

As an example, suppose we consider the preceding model with additional regressors `firmsize`, `msa`, and `injury`. Testing at level 0.05, we obtain

```
. * Stepwise forward selection using statistical significance at 5. stepwise, pe(.05): poisson docvis private chronic female income
>     firmsize msa injury, vce(robust)
Wald test, begin with empty model:
p = 0.0000 < 0.0500, adding chronic
p = 0.0000 < 0.0500, adding injury
p = 0.0000 < 0.0500, adding private
p = 0.0000 < 0.0500, adding female
p = 0.0001 < 0.0500, adding income
Poisson regression                                         Number of obs = 4,412
Log pseudolikelihood = -17503.945                         Wald chi2(5) = 857.27
                                                               Prob > chi2 = 0.0000
                                                               Pseudo R2 = 0.2366



|         | Robust      |           |       |       |                      |           |
|---------|-------------|-----------|-------|-------|----------------------|-----------|
| docvis  | Coefficient | std. err. | z     | P> z  | [95% conf. interval] |           |
| chronic | .9663314    | .0566569  | 17.06 | 0.000 | .8552858             | 1.077377  |
| injury  | .7503917    | .0650373  | 11.54 | 0.000 | .6229209             | .8778626  |
| private | .791386     | .106635   | 7.42  | 0.000 | .5823854             | 1.000387  |
| female  | .528642     | .0567367  | 9.32  | 0.000 | .41744               | .6398439  |
| income  | .0042092    | .001076   | 3.91  | 0.000 | .0021003             | .0063182  |
| _cons   | -.4054209   | .1079598  | -3.76 | 0.000 | -.6170182            | -.1938236 |


```

The preferred model includes `injury` but not `firmsize` and `msa`.

Backward selection, a general-to-specific approach, starts with the most general model and sequentially drops the least statistically significant regressor, provided this regressor is statistically insignificant at the prespecified significance level.

The `pr()` option of the `stepwise` prefix implements backward selection. The option `hierarchical` of the `stepwise` prefix implements forward or backward selection in order of the specified regressors.

### 11.3.8 Pretest bias

The inclusion of variables on the basis of statistical significance leads to so-called pretest bias of the OLS estimator.

Suppose the true model is  $y = \beta_1 + \beta_2 x + u$ , where  $\beta_2 \neq 0$ ,  $x$  is nonstochastic, and  $u \sim N(0, \sigma^2)$ . Then, given a random sample of size  $N$ , the OLS estimator from regression of  $y$  on an intercept and  $x$  is unbiased, so  $E(\hat{\beta}_2) = \beta_2$ .

Suppose instead we first test for statistical significance of  $x$  and include  $x$  in the model only if  $x$  is statistically significant at level 0.05, that is, if  $|t| = |\hat{\beta}_2/s_{\hat{\beta}_2}| > t_{.025}(N - 2)$ . Then this estimator  $\tilde{\beta}_2$  equals  $\hat{\beta}_2$  with probability less than one and equals zero with probability greater than zero. It follows that  $\tilde{\beta}_2$  is biased for  $\beta_2$ , with bias toward zero.

Going the other way, if  $\beta_2 = 0$ , then 5% of the time  $x$  will be erroneously included as a regressor. This problem is compounded when many potential regressors are considered, and test  $p$ -values should be appropriately adjusted; see section [11.6](#) on multiple testing.

To avoid such complications, microeconomics studies commonly minimize pretesting. Instead, models include many regressors, suggested by economic theory or previous studies, or both, regardless of whether they are statistically significant.

[Kozbur \(2020\)](#) proposes a testing-based method for sequentially selecting regressors until a stopping rule is reached. Machine learning methods such as the lasso include as regressors those variables that best predict  $y$ , and some methods do so in ways that avoid the complication of pretest bias; see section 28.8.

### 11.3.9 Wald confidence intervals

Stata output provides Wald confidence intervals for individual regression parameters  $\beta_j$  of the form  $\hat{\beta}_j \pm z_{\alpha/2} \times s_{\hat{\beta}_j}$ , where  $z_{\alpha/2}$  is a standard normal critical value. For some linear-model commands, the critical value is from the  $t$  distribution rather than the standard normal. The default is a 95% confidence interval, which is  $\hat{\beta}_j \pm 1.96 \times s_{\hat{\beta}_j}$  if standard normal critical values (with  $\alpha = 0.05$ ) are used. This default can be changed in Stata estimation commands by using the `level()` option, or it can be changed globally by using the `set level` command.

Now consider any scalar, say,  $\gamma$ , that is a function  $g(\boldsymbol{\beta})$  of  $\boldsymbol{\beta}$ . Examples include  $\gamma = \beta_2$ ,  $\gamma = \beta_2 + \beta_3$ , and  $\gamma = \beta_2/\beta_3$ . A Wald  $100(1 - \alpha)\%$  confidence interval for  $\gamma$  is

$$\hat{\gamma} \pm z_{\alpha/2} \times s_{\hat{\gamma}} \quad (11.6)$$

where  $\hat{\gamma} = g(\hat{\beta})$ , where by  $z_{\alpha/2}$  we mean the area in the right tail is  $\alpha/2$  and  $s_{\hat{\gamma}}$  is the standard error of  $\hat{\gamma}$ . For the nonlinear estimator  $\hat{\beta}$ , the critical value  $z_{\alpha/2}$  is usually used, and for the linear estimator, the critical value  $t_{\alpha/2}$  is usually used. Implementation requires computation of  $\hat{\gamma}$  and  $s_{\hat{\gamma}}$ , using (11.7) and (11.8) given below.

### 11.3.10 The lincom command

The `lincom` command calculates the confidence interval for a scalar linear combination of the parameters  $\mathbf{R}\beta - r$ . The syntax is

```
lincom exp [ , options ]
```

The `eform` option reports exponentiated coefficients, standard errors, and confidence intervals. This is explained in section [11.3.12](#).

The confidence interval is computed by using (11.6), with  $\hat{\gamma} = \mathbf{R}\hat{\beta} - r$  and the squared standard error

$$s_{\hat{\gamma}}^2 = \mathbf{R}\hat{V}(\hat{\beta})\mathbf{R}' \quad (11.7)$$

We consider a confidence interval for  $\beta_{\text{private}} + \beta_{\text{chronic}} - 1$ . We have

```
. * Confidence interval for linear combinations using lincom
. qui use mus210mepsdocvisyoung, clear
. qui keep if year02==1
. qui poisson docvis private chronic female income if year02==1, vce(robust)
. lincom private + chronic - 1
( 1) [docvis]private + [docvis]chronic = 1
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
(1)	.8905303	.1184395	7.52	0.000	.6583932 1.122668

The 95% confidence interval is  $[0.66, 1.12]$  and is based on standard normal critical values because we used the `lincom` command after `poisson`. If instead it had

followed `regress`, then  $t(N - K)$  critical values would have been used.

The `lincom` command also provides a test statistic and  $p$ -value for the two-sided test of  $H_0: \beta_{\text{private}} + \beta_{\text{chronic}} - 1 = 0$ . Then

$z^2 = 7.52^2 \approx (0.8905303/0.1184395)^2 = 56.53$ , which equals the  $W$  obtained in section [11.3.3](#) in the example using `test`, `mtest`. The `lincom` command enables a one-sided test because, unlike using  $W$ , we know the sign of  $z$ .

### 11.3.11 The `nlcom` command (delta method)

The `nlcom` command calculates the confidence intervals in [\(11.6\)](#) for a scalar nonlinear function  $g(\boldsymbol{\beta})$  of the parameters. The syntax is

```
nlcom [name:] exp [, options]
```

The confidence interval is computed by using [\(11.6\)](#), with  $\hat{\gamma} = g(\hat{\boldsymbol{\beta}})$  and the squared standard error

$$s_{\hat{\gamma}}^2 = \partial\gamma/\partial\boldsymbol{\theta} \Big|_{\hat{\boldsymbol{\theta}}} \hat{V}(\hat{\boldsymbol{\theta}}) \partial\gamma/\partial\boldsymbol{\theta}' \Big|_{\hat{\boldsymbol{\theta}}} \quad (11.8)$$

The standard error  $s_{\hat{\gamma}}$  and the resulting confidence interval are said to be computed by the delta method because of the derivative  $\partial\gamma/\partial\boldsymbol{\theta}$ .

As an example, consider confidence intervals for  $\gamma = \beta_{\text{female}}/\beta_{\text{private}} - 1$ . We have

```
. * Confidence interval for nonlinear function of parameters using nlcom
. nlcom _b[female] / _b[private] - 1
    _nl_1: _b[female] / _b[private] - 1
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
_nl_1	-.383286	.1042734	-3.68	0.000	-.587658 -.1789139

Note that  $z^2 = (-3.68)^2 \approx (-0.383286/0.1042734)^2 = 13.51$ . This equals the  $W$  for the test of  $H_0: \beta_{\text{female}}/\beta_{\text{private}} - 1$  obtained by using the `testnl` command in section [11.3.6](#).

### 11.3.12 Asymmetric confidence intervals

For several nonlinear models, such as those for binary outcomes and durations, interest often lies in exponentiated coefficients that are given names such as hazard ratio or odds ratio depending on the application. In these cases, we need a confidence interval for  $e^{\hat{\beta}}$  rather than  $\hat{\beta}$ . This can be done by using either the `lincom` command with the `eform` option or the `nlcom` command. These methods lead to different confidence intervals, with the former preferred.

We can directly obtain a 95% confidence interval for  $\exp(\beta_{\text{private}})$ , using the `lincom, eform` command. We have

```
. * Confidence interval for exp(b) using lincom option eform
. lincom private, eform
( 1) [docvis]private = 0
```

docvis	exp(b)	Std. err.	z	P> z	[95% conf. interval]
(1)	2.222572	.2422636	7.33	0.000	1.795038 2.751935

This confidence interval is computed by first obtaining the usual 95% confidence interval for  $\beta_{\text{private}}$  and then exponentiating the lower and upper bounds of the interval. We have

```
. * Confidence interval for exp(b) using lincom followed by exponentiate
. lincom private
( 1) [docvis]private = 0
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
(1)	.7986652	.1090014	7.33	0.000	.5850263 1.012304

Because  $\beta_{\text{private}} \in [0.5850, 1.0123]$ , it follows that  $\exp(\beta_{\text{private}}) \in [e^{0.5850}, e^{1.0123}]$ , so  $\exp(\beta_{\text{private}}) \in [1.795, 2.752]$ , which is the interval given by `lincom, eform`.

If instead we use `nlcom`, we obtain

```
. * Confidence interval for exp(b) using nlcom
. nlcom exp(_b[private])
_nl_1: exp(_b[private])
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
_nl_1	2.222572	.2422636	9.17	0.000	1.747744 2.6974

The interval is instead  $\exp(\beta_{\text{private}}) \in [1.748, 2.697]$ . This differs from the [1.795, 2.752] interval obtained with `lincom`, and the difference between the two methods can be much larger in other applications.

Which interval should we use? The two are asymptotically equivalent but can differ considerably in small samples. The interval obtained by using `nlincom` is symmetric about  $\exp(\hat{\beta}_{\text{private}})$  and could include negative values (if  $\hat{\beta}$  is small relative to  $s_{\hat{\beta}}$ ). The interval obtained by using `lincom, eform` is asymmetric and, necessarily, is always positive because of exponentiation. This is preferred.

### 11.3.13 Confidence intervals from inverting a test statistic

A confidence interval can be obtained by inverting a test. Specifically, to obtain a 95% confidence interval for  $\beta$ , we perform a two-sided test of  $\beta = \beta^*$  for a range of values of  $\beta^*$ . The confidence interval is then those values of  $\beta^*$  for which the test had  $p > 0.05$  because the 95% confidence interval includes those values that we do not reject at level 0.05.

To illustrate this method, we perform a series of Wald tests to obtain a 95% Wald confidence interval for  $\beta_{\text{private}}$ , using a grid search over null hypothesis values of  $\beta^*$  ranging from -2.0 to 2.0. We obtain

```
. * Confidence interval from inverting test statistic
. qui poisson docvis private chronic female income, vce(robust) nolog
. postfile cifromtest b2 pvalue using pvalues, replace
(file pvalues.dta not found)
. forvalues i = 1/1000 {
  2.      scalar b2 = (`i' - 500)/250
  3.      qui test _b[private] = b2
  4.      scalar p = r(p)
  5.      post cifromtest (b2) (p)
  6. }
. postclose cifromtest
. use pvalues, clear
. sum if pvalue > 0.05
```

Variable	Obs	Mean	Std. dev.	Min	Max
b2	107	.8	.124129	.588	1.012
pvalue	107	.3968495	.2846993	.050327	.9902298

The resulting 95% confidence interval for  $\beta_{\text{private}}$  is [0.588, 1.012] because tests that  $\beta_{\text{private}} = \beta^*$  had  $p > 0.05$  for  $\beta^*$  between 0.588 and 1.012.

This confidence interval, aside from rounding error, is the same as [0.585, 1.012], the usual Wald confidence interval given earlier in the usual output from the `poisson` command. There is no reason to use this method if a Wald confidence interval is available.

The advantage of this more general approach is that it can be used in a wider range of settings. For example, if the only test available is an LR test, then the same procedure can be used to obtain an LR confidence interval.

In more general cases the confidence interval need not be contiguous. For example, one can obtain confidence intervals such as  $\{[-\infty, 1.2] \cup [5.3, 8.7]\}$ . Confidence intervals obtained using weak instruments asymptotics are obtained by inversion of test statistics and thus can be composed of disjoint intervals.

## 11.4 Likelihood-ratio tests

An alternative to the Wald test is the LR test. This is applicable only to ML estimation, under the assumption that the density is correctly specified.

### 11.4.1 LR test statistic

Let  $L(\boldsymbol{\theta}) = f(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$  denote the likelihood function, and consider testing the  $h$  hypotheses  $H_0: \mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ . Distinguish between the usual unrestricted maximum-likelihood estimator (MLE)  $\hat{\boldsymbol{\theta}}_u$  and the restricted MLE  $\tilde{\boldsymbol{\theta}}_r$  that maximizes the log likelihood subject to the restriction  $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ .

The motivation for the LR test is that if  $H_0$  is valid, then imposing the restrictions in estimation of the parameters should make little difference to the maximized value of the likelihood function. The LR test statistic is

$$\text{LR} = -2 \left\{ \ln L(\tilde{\boldsymbol{\theta}}_r) - \ln L(\hat{\boldsymbol{\theta}}_u) \right\} \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0$$

At the 0.05 level, for example, we reject if  $p = \Pr\{\chi^2(h) > \text{LR}\} < 0.05$  or, equivalently, if  $\text{LR} > \chi^2_{0.05}(h)$ , where by  $\chi^2_{0.05}(h)$  we mean the area in the right tail is 0.05. It is unusual to use an  $F$  variant of this test.

The LR and Wald tests, under the conditions in which `vce(oim)` is specified, are asymptotically equivalent under  $H_0$  and local alternatives, so there is no a priori reason to prefer one over the other.

Nonetheless, the LR test is preferred in fully parametric settings, in part because the LR test is invariant under nonlinear transformations, whereas the Wald test is not, as was demonstrated in section [11.3.6](#).

Microeconometricians use Wald tests more often than LR tests because, wherever possible, fully parametric models are not used. For example, consider a linear regression with cross-sectional data. Assuming normal homoskedastic errors permits the use of an LR test. But the preference is to

relax this assumption, obtain a robust estimate of the VCE, and use this as the basis for Wald tests.

The LR test requires fitting two models, whereas the Wald test requires fitting only the unrestricted model. And restricted ML estimation is not always possible. The Stata ML commands can generally be used with the `constraint()` option, but this supports only linear restrictions on the parameters.

Stata output for ML estimation commands uses LR tests in two situations: first, to perform tests on a key auxiliary parameter; and second, in the test for joint significance of regressors automatically provided as part of Stata output, if the default `vce(oim)` option is used.

We demonstrate this for negative binomial regression, a generalization of Poisson regression, for doctor visits by using default ML standard errors. We have

```
. * LR tests output if estimate by ML with default estimate of VCE
. qui use mus210mepsdocvisyoung, clear
. qui keep if year02==1
. nbreg docvis private chronic female income, nolog
Negative binomial regression                                         Number of obs = 4,412
Dispersion: mean                                                 LR chi2(4)    = 1067.55
Log likelihood = -9855.1389                                         Prob > chi2   = 0.0000
                                                               Pseudo R2   = 0.0514
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.8876559	.0594232	14.94	0.000	.7711886 1.004123
chronic	1.143545	.0456778	25.04	0.000	1.054018 1.233071
female	.5613027	.0448022	12.53	0.000	.473492 .6491135
income	.0045785	.000805	5.69	0.000	.0030007 .0061563
_cons	-.4062135	.0611377	-6.64	0.000	-.5260411 -.2863858
/lnalpha	.5463093	.0289716			.4895261 .6030925
alpha	1.726868	.05003			1.631543 1.827762

LR test of alpha=0: chibar2(01) = 1.7e+04 Prob >= chibar2 = 0.000

Here the overall test for joint significance of the four coefficients, given as  $\text{LR chi2}(4) = 1067.55$ , is an LR test.

The last line of output provides an LR test of  $H_0: \alpha = 0$  against  $H_a: \alpha > 0$ . Rejection of  $H_0$  favors the more general negative binomial model because the Poisson is the special case  $\alpha = 0$ . This LR test is nonstandard because the null hypothesis is on the boundary of the parameter space (the negative binomial model restricts  $\alpha \geq 0$ ). In this case, the LR statistic has a distribution that has a probability mass of 1/2 at 0 and a half- $\chi^2(1)$  distribution above 0. This distribution is known as the chibar-0-1 distribution and is used to calculate the reported  $p$ -value of 0.000, which strongly rejects the Poisson in favor of the negative binomial model.

More generally, an LR test of more than one hypothesis may have the null hypothesis on the boundary of the parameter space. For example, this is the case for tests of the joint statistical significance of variance components in mixed models fit using the `mixed` and `me` commands. Then the distribution of the LR test statistic is unknown. Stata computes  $p$ -values using the usual  $\chi^2(h)$  distribution. This leads to a conservative test, one less likely to reject the null hypothesis, because the true  $p$ -value, while not precisely known, is at least known to be smaller than that obtained using the  $\chi^2(h)$  distribution.

### 11.4.2 The lrtest command

The `lrtest` command calculates an LR test of one model that is nested in another when both are fit by using the same ML command. The syntax is

```
lrtest modelspec1 [modelspec2] [, options]
```

where ML results from the two models have been saved previously by using `estimates store` with the names `modelspec1` and `modelspec2`. The order of the two models does not matter. The variation `lrtest modelspec1` requires applying `estimates store` only to the model other than the most recently fitted model.

We perform an LR test of  $H_0: \beta_{\text{private}} = 0, \beta_{\text{chronic}} = 0$  by fitting the unrestricted model with all regressors and then fitting the restricted model with `private` and `chronic` excluded. We fit a negative binomial model because this is a reasonable parametric model for these overdispersed count data, whereas the Poisson was strongly rejected in the test of  $H_0: \alpha = 0$  in the previous section. We have

```

. * LR test using command lrtest
. qui nbreg docvis private chronic female income
. estimates store unrestrict
. qui nbreg docvis female income
. estimates store restrict
. lrtest unrestrict restrict

Likelihood-ratio test
Assumption: restrict nested within unrestrict

LR chi2(2) = 808.74
Prob > chi2 = 0.0000

```

The null hypothesis is strongly rejected because  $p = 0.000$ . We conclude that `private` and `chronic` should be included in the model.

The same test can be performed with a Wald test. To be comparable with the LR test, we use default standard errors here, though in practice the Wald test is used following estimation with robust standard errors.

```

. * Wald test of the same hypothesis and using default standard errors (like LR)
. qui nbreg docvis private chronic female income
. test chronic private
( 1) [docvis]chronic = 0
( 2) [docvis]private = 0
chi2( 2) = 852.26
Prob > chi2 = 0.0000

```

The results differ somewhat, with test statistics of 809 and 852. The differences in these asymptotically equivalent tests can be considerably larger in other applications, especially those with few observations.

### 11.4.3 Direct computation of LR tests

The default is for the `lrtest` command to compute the LR test statistic only in situations where it is clear that the LR test is appropriate. The command will produce an error when, for example, the `vce(robust)` option is used or when different estimation commands are used. The `force` option causes the LR test statistic to be computed in such settings, with the onus on the user to verify that the test is still appropriate.

As an example, we return to the LR test of Poisson against the negative binomial model, automatically given after the `nbreg` command, as discussed in section [11.4.1](#). To perform this test using the `lrtest` command, we need the `force` option because two different estimation commands, `poisson` and `nbreg`, are used. We have

```
. * LR test using option force
. qui nbreg docvis private chronic female income
. estimates store nb
. qui poisson docvis private chronic female income
. estimates store poiss
. lrtest nb poiss, force
Likelihood-ratio test
Assumption: poiss nested within nb
LR chi2(1) = 17296.82
Prob > chi2 = 0.0000
. display "Corrected p-value for LR-test = " r(p)/2
Corrected p-value for LR-test = 0
```

As expected, the LR statistic is the same as  $\text{chibar2}(01) = 1.7e+04$ , reported in the last line of output from `nbreg` in section [11.4.1](#). The `lrtest` command automatically computes  $p$ -values using  $\chi^2(h)$ , where  $h$  is the difference in the number of parameters in the two fitted models, here  $\chi^2(1)$ . As explained in section [11.4.1](#), however, half- $\chi^2(1)$  should be used in this particular example, providing a cautionary note for the use of the `force` option.

#### 11.4.4 Tests at the boundary

As noted in section [11.4.1](#), in some cases, the range of a parameter is restricted, and a test is performed of whether the parameter takes the value at its boundary. In particular, the parameter of interest may be a variance that is by definition restricted to be nonnegative, and we may want to test whether it equals zero. In such cases, the distribution of the LR statistic for tests of  $q$  restrictions is no longer the  $\chi^2(q)$  distribution.

For tests of a single restriction, the LR statistic has a half- $\chi^2(1)$  distribution, which Stata output labels `chibar2(01)`. This distribution can be used in the usual way for  $p < 0.50$ .

For  $q > 1$ , the actual distribution is quite complicated. Instead, the  $\chi^2(q)$  distribution continues to be used. This provides a conservative test. For example, if the reported  $p = 0.07$ , then the true  $p < 0.07$ . A common example is a joint hypothesis test on the variances and covariances in a mixed model such as that in section [6.7.6](#).

## 11.5 Lagrange multiplier test (or score test)

The third major hypothesis testing method is a test method usually referred to as the score test by statisticians and as the LM test by econometricians. This test is less often used, aside from some leading model-specification tests in situations where the null hypothesis model is easy to fit but the alternative hypothesis model is not.

### 11.5.1 LM tests

The unrestricted MLE  $\hat{\boldsymbol{\theta}}_u$  sets  $\mathbf{s}(\hat{\boldsymbol{\theta}}_u) = \mathbf{0}$ , where  $\mathbf{s}(\boldsymbol{\theta}) = \partial \ln L(\boldsymbol{\theta}) / \partial \boldsymbol{\theta}$  is called the score function. An LM test, or score test, is based on closeness of  $\mathbf{s}(\tilde{\boldsymbol{\theta}}_r)$  to zero, where evaluation is now at  $\tilde{\boldsymbol{\theta}}_r$ , the alternative restricted MLE that maximizes  $\ln L(\boldsymbol{\theta})$  subject to the  $h$  restrictions  $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ . The motivation is that if the restrictions are supported by the data, then  $\tilde{\boldsymbol{\theta}}_r \simeq \hat{\boldsymbol{\theta}}_u$ , so  $\mathbf{s}(\tilde{\boldsymbol{\theta}}_r) \simeq \mathbf{s}(\hat{\boldsymbol{\theta}}_u) = \mathbf{0}$ .

Because  $\mathbf{s}(\tilde{\boldsymbol{\theta}}_r) \stackrel{a}{\sim} N\{\mathbf{0}, \text{Var}(\tilde{\boldsymbol{\theta}}_r)\}$ , we form a quadratic form that is a chi-squared statistic, similar to the method in section [11.3.1](#). This yields the LM test statistic, or score test statistic, for  $H_0: \mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ :

$$\text{LM} = \mathbf{s}(\tilde{\boldsymbol{\theta}}_r)' \left[ \widehat{V} \left\{ \mathbf{s}(\tilde{\boldsymbol{\theta}}_r) \right\} \right]^{-1} \mathbf{s}(\tilde{\boldsymbol{\theta}}_r) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0$$

At the 0.05 level, for example, we reject if  $p = \Pr\{\chi^2(h) > \text{LM}\} < 0.05$  or, equivalently, if  $\text{LM} > \chi^2_{0.05}(h)$ . It is not customary to use an  $F$  variant of this test.

The preceding motivation explains the term “score test”. The test is also called the LM test for the following reason: Let  $\ln L(\boldsymbol{\theta})$  be the log-likelihood function in the unrestricted model. The restricted MLE  $\tilde{\boldsymbol{\theta}}_r$  maximizes  $\ln L(\boldsymbol{\theta})$  subject to  $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ , so  $\tilde{\boldsymbol{\theta}}_r$  maximizes  $\ln L(\boldsymbol{\theta}) - \lambda' \mathbf{g}(\boldsymbol{\theta})$ . An LM test is based on whether the associated LMS  $\tilde{\boldsymbol{\lambda}}_r$  of this restricted optimization are close to

0 because  $\lambda = \mathbf{0}$  if the restrictions are valid. It can be shown that  $\tilde{\lambda}_r$  is a full-rank matrix multiple of  $\mathbf{s}(\tilde{\theta}_r)$ , so the LM and score tests are equivalent.

Under the conditions in which `vce(oim)` is specified, the LM test, LR test, and Wald test are asymptotically equivalent for  $H_0$  and local alternatives, so there is no a priori reason to prefer one over the others. The attraction of the LM test is that, unlike Wald and LR tests, it requires fitting only the restricted model. This is an advantage if the restricted model is easier to fit, such as a homoskedastic model rather than a heteroskedastic model. Furthermore, an asymptotically equivalent version of the LM test can often be computed by the use of an auxiliary regression. On the other hand, there is generally no universal way to implement an LM test, unlike Wald and LR tests. If the LM test rejects the restrictions, we then still need to fit the unrestricted model.

### 11.5.2 The `estat` commands

Because LM tests are estimator specific and model specific, there is no `lmtest` command. Instead, LM tests usually appear as `estat` postestimation commands to test misspecifications.

A leading example is the `estat hettest` command to test for heteroskedasticity after `regress`. This LM test is implemented by auxiliary regression, which is detailed in section 3.7. The default version of the test requires that under the null hypothesis, the independent homoskedastic errors must be normally distributed, whereas the `iid` option relaxes the normality assumption to one of independent and identically distributed errors.

Another example is the `xttest0` command to implement an LM test for random effects after `xtreg`. Yet another example is the LM test for overdispersion in the Poisson model, given in an end-of-chapter exercise.

### 11.5.3 LM test by auxiliary regression

For ML estimation with a correctly specified density, an asymptotically equivalent version of the LM statistic can always be obtained from the following auxiliary procedure. First, obtain the restricted MLE  $\tilde{\theta}_r$ . Second,

form the scores for each observation of the unrestricted model,  $s_i(\boldsymbol{\theta}) = \partial \ln f(y_i | \mathbf{x}_i, \boldsymbol{\theta}) / \partial \boldsymbol{\theta}$ , and evaluate them at  $\tilde{\boldsymbol{\theta}}_r$  to give  $s_i(\tilde{\boldsymbol{\theta}}_r)$ . Third, compute  $N$  times the uncentered  $R^2$  (or, equivalently, the model sum of squares) from the auxiliary regression of 1 on  $s_i(\tilde{\boldsymbol{\theta}}_r)$ .

It is easy to obtain restricted model scores evaluated at the restricted MLE or unrestricted model scores evaluated at the unrestricted MLE. However, this auxiliary regression requires computation of the unrestricted model scores evaluated at the restricted MLE. If the parameter restrictions are linear, then these scores can be obtained by using the `constraint` command to define the restrictions before estimation of the unrestricted model.

We illustrate this method for the LM test of whether  $H_0: \beta_{\text{private}} = 0, \beta_{\text{chronic}} = 0$  in a negative binomial model for `docvis` that, when unrestricted, includes as regressors an intercept, `female`, `income`, `private`, and `chronic`. The restricted MLE  $\tilde{\boldsymbol{\beta}}_r$  is then obtained by negative binomial regression of `docvis` on all these regressors, subject to the constraint that  $\beta_{\text{private}} = 0$  and  $\beta_{\text{chronic}} = 0$ . The two constraints are defined by using the `constraint` command, and the restricted estimates of the unrestricted model are obtained using the `nbreg` command with the `constraints()` option. Scores can be obtained by using the `predict` command with the `scores` option. However, these scores are derivatives of the log density with respect to model indices (such as  $\mathbf{x}'_i \boldsymbol{\beta}$ ) rather than with respect to each parameter. Thus, following `nbreg` only two “scores” are given,  $\partial \ln f(y_i) / \partial \mathbf{x}'_i \boldsymbol{\beta}$  and  $\partial \ln f(y_i) / \partial \alpha$ . These two scores are then expanded to  $K + 1$  scores  $\partial \ln f(y_i) / \partial \beta_j = \{\partial \ln f(y_i) / \partial \mathbf{x}'_i \boldsymbol{\beta}\} \times x_{ij}, j = 1, \dots, K$ , where  $K$  is the number of regressors in the unrestricted model, and  $\partial \ln f(y_i) / \partial \alpha$ , where  $\alpha$  is the scalar overdispersion parameter. Then, 1 is regressed on these  $K + 1$  scores.

We have

```

. * Perform LM test that b_private=0, b_chronic=0 using auxiliary regression
. qui use mus210mepsdocvisyoung, clear
. qui keep if year02==1
. generate one = 1
. constraint define 1 private = 0
. constraint define 2 chronic = 0
. qui nbreg docvis female income private chronic, constraints(1 2)
. predict eqscore ascore, scores
. generate s1restb = eqscore*one
. generate s2restb = eqscore*female
. generate s3restb = eqscore*income
. generate s4restb = eqscore*private
. generate s5restb = eqscore*chronic
. generate salpha = ascore*one
. qui regress one s1restb s2restb s3restb s4restb s5restb salpha, noconstant
. scalar lm = e(N)*e(r2)
. display "LM = N x uncentered Rsq = " lm " and p = " chi2tail(2,lm)
LM = N x uncentered Rsq = 424.17616 and p = 7.786e-93

```

The null hypothesis is strongly rejected with  $LM = 424$ . By comparison, in section [11.4.2](#), the asymptotically equivalent LR and Wald statistics for the same hypothesis were, respectively, 809 and 852.

The divergence of these purportedly asymptotically equivalent tests is surprising given the large sample size of 4,412 observations. One explanation, always a possibility with real data, is that the unknown data-generating process (DGP) for these data is not the fitted negative binomial model—the asymptotic equivalence holds only under  $H_0$ , which includes correct model specification. A second explanation is that this LM test has poor size properties even in relatively large samples. This explanation could be pursued by adapting the simulation exercise in section [11.7](#) to one for the LM test with data generated from a negative binomial model.

Often, more than one auxiliary regression is available to implement a specific LM test. The easiest way to implement an LM test is to find a reference that defines the auxiliary regression for the example at hand and then implement the regression. For example, to test for heteroskedasticity in the linear regression model that depends on variables  $\mathbf{z}_i$ , we calculate  $N$  times the uncentered explained sum of squares from the regression of

squared OLS residuals  $\hat{u}_i^2$  on an intercept and  $\mathbf{z}_i$ ; all that is needed is the computation of  $\hat{u}_i$ . In this case, `estat hettest` implements this anyway.

The auxiliary regression versions of the LM test are known to have poor size properties, though in principle these can be overcome by using the bootstrap with asymptotic refinement. For example, see [Cribari-Neto and Zarkos \(1999\)](#).

## 11.6 Multiple testing

Standard theory assumes that hypothesis tests are done once only and in isolation. In some social sciences and biological sciences, many experiments on a phenomenon may be run by different researchers, with only studies statistically significant at level 0.05 being published because of publication bias. Then the multiple testing methods presented below are not feasible because of lack of knowledge of the unpublished studies. In such settings, a highly cited study ([Benjamin et al. 2018](#)) has proposed that for claims of new discoveries, the default  $p$ -value for statistical significance should be 0.005. For a standard asymptotic two-sided Wald test, this corresponds to rejection if  $|t| > 2.80$ .

Many empirical studies in economics include multiple tests or multiple comparisons that simultaneously test several hypotheses. Examples include testing the statistical significance of a key regressor in several subgroups of the sample (subgroup analysis); testing the statistical significance of a key regressor in regressions on a range of outcomes (multiple outcomes); and testing the significance of a wide range of variables on a single outcome (multiple regressors), such as which specific genes are related to a particular form of cancer.

Performing many such tests at standard significance levels such as five percent is very likely to lead to spurious statistical significance. For example, in 100 independent tests at level 0.05, we expect on average to obtain 5 statistically significant results, and at least 1 statistically significant result with high probability, even if no effect is present. In such cases, one should view the entire battery of tests as a unit and appropriately adjust the significance levels of individual tests.

Controlling for multiple testing makes it more difficult to reject the null hypothesis. Furthermore, the simplest methods developed for multiple testing tend to be conservative, leading to low power and underrejection of the null hypothesis. The development of less conservative methods is a current area of research.

### 11.6.1 Family-wise error rate

We now consider testing  $m$  hypotheses  $H_i$  against alternative  $H'_i$ ,  $i = 1, \dots, m$ . The FWER is the probability of incorrectly finding statistical significance (a type I error) in at least one of the many tests conducted. The goal is to test each of the individual hypotheses in such a way that the FWER is controlled to be at most  $\alpha$ , so

$$\text{FWER} = \Pr(\text{incorrectly reject at least one hypothesis } H_i) \leq \alpha$$

This form of control, one appropriate for econometric studies, is called strong control and controls the familywise error rate regardless of which null hypotheses are true and which are false. An alternative form of control, called weak control, controls FWER under the additional condition that all  $m$  hypotheses are true.

Several methods have been proposed to determine the level of each individual test that leads to a test with a desired FWER  $\alpha$ .

A correction that can always be used is the Bonferroni correction that performs each individual test at level  $\alpha_i = \alpha/m$ .

For example, if  $m = 10$  tests are conducted with FWER = 0.05, then each test should be at level  $\alpha_i = 0.05/10 = 0.005$ . The motivation is Bonferroni's inequality that  $\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$ , so if two tests are performed, each at level  $\alpha/2$ , then the probability that at least one of the two tests rejects is at most  $\alpha/2 + \alpha/2 = \alpha$ . The Bonferroni correction leads to conservative tests because it provides an upper bound to the FWER.

Holm's step-down procedure provides a less conservative refinement of the Bonferroni correction. The  $m$  individual  $p$ -values are ordered in increasing value, say,  $p_i, i = 1, \dots, m$ . Then, using Bonferroni's method, we reject the hypothesis corresponding to the lowest  $p$ -value if  $p_1 < \alpha/m$ . Given rejection, there are then  $(m - 1)$  remaining hypotheses to simultaneously test, so the next test rejects if  $p_2 < \alpha/(m - 1)$ , and so on. Thus, we test at increasing test levels  $\alpha_i = \alpha/(m - i + 1)$ , stopping when  $p_i > \alpha_i$ .

Consider the following example with  $\alpha = 0.05$  and  $m = 8$ . The second row gives the ordered test levels  $\alpha_i = 0.05/(9 - i)$ , and we suppose the

ordered  $p$ -values are those given in the third row.

$i$	1	2	3	4	5	6	7	8
$\alpha$	0.00625	0.00714	0.00833	0.010	0.0125	0.01667	0.025	0.05
$p$	0.002	0.004	0.008	0.012	0.015	0.016	0.034	0.058

Then we reject the three hypotheses corresponding to the three smallest  $p$ -values because  $p_1 < \alpha_1$ ,  $p_2 < \alpha_2$ ,  $p_3 < \alpha_3$ , but  $p_4 = 0.012 > \alpha_4 = 0.010$ . The individual tests use the corrected critical  $p$ -value of  $\alpha_4 = 0.010$ , compared with 0.005 using Bonferroni's method.

Hochberg's step-up method is a variant of Holm's method that begins testing with the highest  $p$ -value. We reject the hypothesis corresponding to the largest  $p$ -value if  $p_m < \alpha$ . If we do not reject, then we test using the second-largest  $p$ -value and reject if  $p_{m-1} < \alpha/2$ , and so on. Thus, we compare  $p$ -values to decreasing test levels  $\alpha_{m-i} = \alpha/(m - i + 1)$ . This procedure has greater power than the Holm procedure but is not valid in all circumstances. It is valid if the tests are independent or if the test statistics have a distribution with multivariate positivity of order 2, where  $X_1$  and  $X_2$  are positive dependent if  $\Pr(X_1 \cup X_2) > \Pr(X_1) \times \Pr(X_2)$ .

Applying Hochberg's method to the preceding example, we see the first two tests do not reject, because  $p_8 > \alpha_8$  and  $p_7 > \alpha_7$ , while the third test rejects because  $p_6 = 0.016 < \alpha_6 = 0.01667$ . The individual tests use the corrected critical  $p$ -value of  $\alpha_6 = 0.01667$ , and we reject the six hypotheses with the lowest six  $p$ -values.

By making additional assumptions, one can obtain less conservative tests. In the case of  $m$  statistically independent tests, each at level  $\alpha^*$ , the FWER is exactly  $\alpha = 1 - (1 - \alpha^*)^m$  because the probability under the null hypothesis of finding no statistical significance in all  $m$  tests is  $(1 - \alpha^*)^m$ . Going the other way, if we wish the FWER to equal  $\alpha$ , then each of the  $m$  tests should be at level  $\alpha^* = 1 - (1 - \alpha)^{1/m}$ , called the Šidák correction. For example, if  $m = 10$  tests are conducted with FWER = 0.05, then each test should be at level  $\alpha^* = 1 - 0.95^{1/10} = 0.00512$ , compared with 0.005 for the Bonferroni correction. The Šidák correction is exact under independence of the tests and

is conservative if tests are nonnegatively mutually correlated. But it may overreject if tests are negatively correlated. The Šidák adjustment can be applied to some of the other methods, such as Holm's step-down method. The same  $\alpha$  values can be used to form confidence intervals. For example, consider Šidák's method with  $m = 10$  and  $\alpha = 0.05$  so  $\alpha^* = 0.00512$  and  $1 - \alpha^* = 0.99478$ . An adjusted confidence interval following an estimation command can be implemented using the option `level(99.48)`, where at most two decimal digits can be given in the `level()` option.

The methods can also be used to adjust  $p$ -values. Let  $p$  be the usual  $p$ -value from regression output. Then the Bonferroni-adjusted  $p$ -value from  $m$  tests is the minimum of  $m \times p$  and 1, while the Šidák-adjusted  $p$ -value is  $1 - (1 - p)^m$  and is always less than 1. For example, if  $m = 10$  and an individual test has  $p = 0.04$ , then the Bonferroni-adjusted  $p$ -value is  $\min(1, 0.40) = 0.40$ , while the Šidák-adjusted  $p$ -value is  $1 - 0.96^{10} = 0.335$ .

The Bonferroni and Holm methods can be adjusted to control  $k$ -FWER, the probability of rejecting  $k$  or more of the  $m$  hypotheses.

The `test` command has an option `mtest` that computes these variously adjusted  $p$ -values, but this is limited to tests of individual regressors in a single-equation regression or to tests across equations in jointly estimated equations such as tests following the `sur` command.

The community-contributed `multproc` command ([Newson and the ALSPAC Study Team 2003](#)) performs a wide range of multiple testing procedures that control the FWER, as well as the FDR. It requires as inputs the uncorrected  $p$ -values from individual tests. [Newson and the ALSPAC Study Team \(2003\)](#) provide a summary of the methods used.

## 11.6.2 Subgroup analysis

We continue with the Poisson regression example of doctor visits. We now break the sample into four 10-year age groups and seek to determine in which of the age subgroups private insurance is statistically significant at 5%, controlling for the multiple testing.

The following code creates four age subgroups in age ranges 25–34, 35–44, 45–54, and 55–64. For illustrative purposes, the first 500 observations are used as the  $p$ -values for income become much smaller using all observations.

```
. * Multiple tests - form the age subgroups 25-34, 35-44, 45-54, 55-64
. qui use mus210mepsdocvisyoung, clear
. keep if _n <= 500
(29,624 observations deleted)
. generate agegroup = round(age)-2 // Age is in tens of years
. tabulate agegroup
```

agegroup	Freq.	Percent	Cum.
1	190	38.00	38.00
2	153	30.60	68.60
3	107	21.40	90.00
4	50	10.00	100.00
Total	500	100.00	

The saved output from the `poisson` command does not include  $p$ -values, so we manually compute these using the standard normal distribution. The Bonferroni correction is then  $\min(1, 4p)$  and the Šidák correction is  $1 - (1 - p)^4$ .

```
. * Multiple tests for multiple subgroups using p-value corrections
. global nummodels 4
. forvalues i = 1/$nummodels {
    2.     qui poisson docvis private chronic female income
>         if agegroup==`i', vce(robust)
    3.     scalar beta = _b[private]
    4.     scalar p = 2*(1-normal(abs(_b[private])/_se[private])))
    5.     scalar pbonferroni = min(1,p*$nummodels)
    6.     scalar psidak = 1-(1-p)^$nummodels
    7.     di "Group " `i' " b =" %7.4f beta " p-values:" " Usual =" %6.3f p
>         " Bonferroni =" %6.3f pbonferroni " Sidak =" %6.3f psidak
    8. }
Group 1 b = 0.8409 p-values: Usual = 0.032 Bonferroni = 0.128 Sidak = 0.122
Group 2 b = 1.3040 p-values: Usual = 0.000 Bonferroni = 0.002 Sidak = 0.002
Group 3 b = 0.0530 p-values: Usual = 0.919 Bonferroni = 1.000 Sidak = 1.000
Group 4 b = 1.1587 p-values: Usual = 0.024 Bonferroni = 0.096 Sidak = 0.093
```

Using the usual unadjusted  $p$ -values, we see private insurance appears statistically significant at level 0.05 for those in all age groups except the 45–54 bracket. Once we correct for multiple testing, however, private insurance is statistically significant only in the 35–44 age bracket.

Holm's method yields the same result. The ordered  $p$ -values are 0.000, 0.024, 0.032, and 0.919, while the ordered test levels  $\alpha_i = 0.05/(5 - i)$  are 0.0125, 0.01667, 0.025, and 0.05. We use  $\alpha = 0.01667$  because  $0.000 < 0.0125$  but  $0.024 > 0.01667$ , and only the 45–54 bracket has  $p < 0.01667$ .

#### **multproc community-contributed command**

The community-contributed `multproc` command ([Newson and the ALSPAC Study Team 2003](#)) uses as inputs the  $p$ -values from various tests. Here we apply the command to the preceding four subgroup tests, using the `method(holm)` option to implement Holm's step-down method. The various `method()` options cover 12 different multiple testing methods.

The example demonstrates various options of the command, though only the first three options are essential.

```
. * Multiple tests using community-contributed command multproc and Holm's method
. input pvalues
    pvalues
1.      .032
2.      .000
3.      .919
4.      .024
5. end

. multproc, puncor(0.05) pvalue(pvalues) method(holm) rank(prank)
>     gpuncor(alpha) gpcor(indivalpha) nhcred(credible) reject(reject)

Method: holm
Uncorrected overall critical P-value: .05
Number of P-values: 4
Corrected overall critical P-value: .01666667
Number of rejected P-values: 1

. list pvalues indivalpha reject credible prank alpha in 1/4, clean
    pvalues    indival~a    reject    credible    prank    alpha
1.      .032    .01666667        0        1        3      .05
2.          0    .01666667        1        0        1      .05
3.      .919    .01666667        0        1        4      .05
4.      .024    .01666667        0        1        2      .05
```

As in the preceding results, the corrected critical value is 0.01667 and the null hypothesis of zero coefficient for `private` is rejected in only the second age group.

#### **mtest option of the tests command**

Stata's `test` command with the `mtest` option is intended for use after a single regression, rather than from four separate regressions.

So we combine the four separate regressions into a single regression with regressors that are fully interacted with variable `agegroup`. To ensure a different intercept in each regression, we manually include an intercept (variable `one`) and then add the `noconstant` option. The complicated names for each coefficient in this interacted regression can be obtained in several ways, including using command `matrix list e(b)`. We use the option `mtest(sidak)`, which gives Šidák-corrected *p*-values. These are valid here because the four subgroups are independent of each other.

```

. * Multiple tests for multiple subgroups using mtests option of test
. qui use mus210mepsdocvisyoung, clear
. keep if _n <= 500
(29,624 observations deleted)
. generate agegroup = round(age)-2 // Age is in tens of years
. generate one = 1
. qui poisson docvis i.agegroup#c.(one private chronic female income),
> noconstant vce(robust)
. qui matrix list e(b) // Yields the complicated names for the coefficients
. test ([docvis]:1b.agegroup#c.private) ([docvis]:2.agegroup#c.private=0)
> ([docvis]:3.agegroup#c.private=0) ([docvis]:4.agegroup#c.private=0),
> mtest(s)
( 1) [docvis]1b.agegroup#c.private = 0
( 2) [docvis]2.agegroup#c.private = 0
( 3) [docvis]3.agegroup#c.private = 0
( 4) [docvis]4.agegroup#c.private = 0

```

	chi2	df	p > chi2
(1)	4.62	1	0.1209*
(2)	12.61	1	0.0015*
(3)	0.01	1	1.0000*
(4)	5.18	1	0.0882*
All	22.41	4	0.0002

\* Sidak-adjusted p-values

The Poisson coefficients from this combined regression, not reported, are exactly the same as those obtained from running separate regressions for each of the four subgroups. The robust standard errors differ slightly, because they are computed using a finite-sample correction factor  $N/(N - K)$  that equals  $500/(500 - 20)$  for the combined sample and, for example,  $190/(190 - 5)$  in the youngest age group. Thus, the Šidák-corrected *p*-values are slightly different from those obtained earlier.

### 11.6.3 Multiple outcomes

As a pedagogical example using the same dataset, we consider the outcomes private insurance, whether injured, and years of schooling. For code brevity, all three models are fit by OLS, though in practice one would use, for example, Poisson, logit, and OLS for these three different types of outcomes

```

. * Multiple tests for multiple outcomes using p-value corrections
. global nummodels 3
. global ylist docvis injury educ
. foreach var of varlist $ylist {
2.     qui regress `var' private chronic female income, vce(robust)
3.     scalar p = 2*ttail(e(df_r),abs(_b[income]/_se[income]))
4.     scalar pbonferroni = min(1,p*$nummodels)
5.     scalar psidak = 1-(1-p)^$nummodels
6.     di "Outcome = " "`var'" _col(19) " p-values:" " Usual =" %7.4f p
>         " Bonferroni =" %7.4f pbonferroni " Sidak =" %7.4f psidak
7. }
Outcome = docvis      p-values: Usual = 0.2334 Bonferroni = 0.7003 Sidak = 0.5495
Outcome = injury       p-values: Usual = 0.0314 Bonferroni = 0.0943 Sidak = 0.0914
Outcome = educ         p-values: Usual = 0.0000 Bonferroni = 0.0000 Sidak = 0.0000

```

Again, the *p*-values can become much larger when we adjust test size for multiple testing, and `income` is statistically significant in only the `educ` regression if the FWER is set to 0.05.

And again, the community-contributed `multproc` command could be used to implement various multiple testing procedures, such as Holm's step-down procedure.

#### 11.6.4 False discovery rate

The FWER controls the probability of erroneously finding statistical significance in even one of the tests conducted. This approach is therefore very stringent when many tests are being conducted, such as determining which of many genes is associated with a particular cancer.

An alternative approach allows a fraction of the tests to erroneously find statistical significance, leading to an increase in test power at the expense of increased test size.

For multiple tests, define the false discovery proportion (FDP) to be the proportion of rejected hypotheses that are falsely rejected, so

$$\text{FDP} = F/R$$

where  $F$  = number of false rejections,  $R$  = total number of rejections, and  $\text{FDP} = 0$  if  $R = 0$ . One possible approach controls the probability that the FDP exceeds a prespecified fraction  $\gamma$ , where  $\gamma = 0$  corresponds to controlling the FWER.

[Benjamini and Hochberg \(1995\)](#) instead proposed controlling the FDR, which is defined to be the expected FDP; that is,

$$\text{FDR} = E(\text{FDP})$$

Their method orders tests by  $p$ -value from smallest to largest, so  $p_1 < p_2 < \dots < p_m$ . We then reject the corresponding ordered hypotheses  $H_1, \dots, H_k$ , where  $k$  is the largest  $i$  for which  $p_i \leq \text{FDR} \times i/m$  and the FDR is the prespecified FDR for the multiple tests. If the multiple tests are independent, then the size of the test equals FDR.

For example, suppose we perform 100 tests and the desired FDR is 10%. Then  $\text{FDR} = 0.10$ , and the ordered critical values are  $0.001, 0.002, 0.003, \dots$ . If the seventh-ordered  $p$ -value is 0.0065 and the eighth-ordered  $p$ -value is 0.0084, then we would reject the seven hypotheses with the seven lowest  $p$ -values.

The Benjamini–Hochberg procedure assumed that tests are statistically independent. [Benjamini and Yekutieli \(2001\)](#) showed that the test remains valid under some forms of dependence but is then conservative. Furthermore, [Benjamini and Yekutieli \(2001\)](#) proposed a more conservative variation of the method that is valid for any form of dependence.

The community-contributed add-on `multproc` performs a wide range of multiple test procedures that control for FDR, in addition to FWER, presented earlier. [Newson and the ALSPAC Study Team \(2003\)](#) provide a summary of the methods used.

### 11.6.5 More powerful multiple tests

In practice, hypothesis tests are not independent and are often positively correlated. Failure to adjust for this correlation leads to conservative tests with actual test size being much less than the nominal test size and hence leads to a substantial loss of test power. This has led to a range of refinements to multiple testing methods and is still an active area of research.

Many of the improvements in power use resampling methods to account for dependence. [Romano and Wolf \(2005\)](#) provide an overview. [Anderson et al. \(2008\)](#) provide an application of a resampling-based step-down procedure for both FWER and FDR based on adjusted  $p$ -values.

[Farcomeni \(2008\)](#) provides an extensive guide to the multiple testing literature. [List, Shaikh, and Xu \(2019\)](#) consider the special case of experimental data in which units are assigned to treatments and control using simple random sampling. They use a bootstrap-based procedure for multiple testing that asymptotically controls the FWER and has much more power than existing methods because it incorporates information about dependence between the multiple tests. The authors provide a community-contributed command `mhtexp` to implement this method.

## 11.7 Test size and power

We consider computation of the test size and power of a Wald test by Monte Carlo simulation. The goal is to determine whether tests that are intended to reject at, say, a 0.05 level really do reject at a 0.05 level and to determine the power of tests against meaningful parameter values under the alternative hypothesis. This extends the analysis of section [5.6](#), which focused on the use of simulation to check the properties of estimators of parameters and estimators of standard errors. Here we instead focus on inference.

### 11.7.1 Simulation DGP: OLS with chi-squared errors

The DGP is the same as that in section [5.6](#), with data generated from a linear model with skewed errors, specifically,

$$y = \beta_1 + \beta_2 x + u; \quad u \sim \chi^2(1) - 1; \quad x \sim \chi^2(1)$$

where  $\beta_1 = 1$ ,  $\beta_2 = 2$  and the sample size  $N = 150$ . The  $[\chi^2(1) - 1]$  errors have a mean of 0 and a variance of 2 and are skewed.

In each simulation, both  $y$  and  $x$  are redrawn, corresponding to random sampling of individuals. We investigate the size and power of  $t$  tests on  $H_0: \beta_2 = 2$ , the DGP value after OLS regression. The tests are based on default standard errors.

### 11.7.2 Test size

In testing  $H_0$ , we can make the error of rejecting  $H_0$  when  $H_0$  is true. This is called a type I error. The test size is the probability of making this error. Thus,

$$\text{Size} = \Pr(\text{Reject } H_0 | H_0 \text{ true})$$

The reported  $p$ -value of a test is the estimated size of the test. Most commonly, we reject  $H_0$  if the size is less than 0.05.

The most serious error is one of incorrect test size, even asymptotically, because of, for example, the use of inconsistent estimates of standard errors if a Wald test is used. Even if this threshold is passed, a test is said to have poor finite-sample size properties or, more simply, poor finite-sample properties, if the reported  $p$ -value is a poor estimate of the true size. Often, the problem is that the reported  $p$ -value is much lower than the true size, so we reject  $H_0$  more often than we should.

For our example with DGP value of  $\beta_2 = 2$ , we want to use simulations to estimate the size of an  $\alpha$ -level test of  $H_0: \beta_2 = 2$  against  $H_a: \beta_2 \neq 2$ . In section 5.6.2, we did so when  $\alpha = 0.05$  by counting the proportion of simulations that led to rejection of  $H_0$  at a level of  $\alpha = 0.05$ . The estimated size was 0.050 because 50 of the 1,000 simulations led to rejection of  $H_0$ . In general, we do not expect exactly 50 simulations to reject because there is simulation randomness.

A computationally more efficient procedure is to compute the  $p$ -value for the test of  $H_0: \beta_2 = 2$  against  $H_a: \beta_2 \neq 2$  in each of the 1,000 simulations because the 1,000  $p$ -values can then be used to estimate the test size for a range of values of  $\alpha$ , as we demonstrate below. The  $p$ -values were computed in the `chi2data` program defined in section 5.6.1 and returned as the scalar `p2`, but these  $p$ -values were not used in any subsequent analysis of test size. We now do so here.

The simulations in section 5.6 were performed by using the `simulate` command. Here we instead use `postfile` and a `forvalues` loop; the code is for the 1,000 simulations:

```

. * Do 1,000 simulations where each gets p-value of test of b2=2
. set seed 10101
. postfile sim pvalues using pvalues, replace
. forvalues i = 1/1000 {
2.     drop _all
3.     qui set obs 150
4.     qui generate double x = rchi2(1)
5.     qui generate y = 1 + 2*x + rchi2(1)-1
6.     qui regress y x
7.     qui test x = 2
8.     scalar p = r(p)           // p-value for test this simulation
9.     post sim (p)
10. }
. postclose sim

```

The simulations produce 1,000  $p$ -values that range from 0 to 1.

```

. * Summarize the p-value from each of the 1,000 tests
. use pvalues, clear
. summarize pvalues

```

Variable	Obs	Mean	Std. dev.	Min	Max
pvalues	1,000	.5027676	.2836176	.0000208	.9994959

These should actually have a uniform distribution, and giving the command `histogram pvalues` reveals that this is the case.

Given the 1,000 values of `pvalues`, we can find the actual size of the test for any choice of  $\alpha$ . For a test at the  $\alpha = 0.05$  level, we obtain

```

. * Determine size of test at level 0.05
. count if pvalues < .05
50
. display "Test size from 1000 simulations = " r(N)/1000
Test size from 1000 simulations = .05

```

The actual test size equals the nominal size of 0.05, though more generally we expect some discrepancy due to simulation randomness. Furthermore, it is exactly the same value as that obtained in section [5.6.1](#) because the same seed and sequence of commands were used there.

The test size is not estimated exactly because of simulation error. If the true size equals the nominal size of  $\alpha$ , then the proportion of times  $H_0$  is

rejected in  $S$  simulations is a random variable with a mean of  $\alpha$  and a standard deviation of  $\sqrt{\alpha(1 - \alpha)/S} \simeq 0.007$  when  $S = 1000$  and  $\alpha = 0.05$ . When we use a normal approximation, the 95% simulation interval for this simulation is  $[0.036, 0.064]$ , and the value we obtained of 0.050 is within this interval. More precisely, the `ci` command yields an exact binomial confidence interval.

```
. * 95% simulation interval using exact binomial at level 0.05 with S=1000
. ci proportions 1000 50
```

Variable	Obs	Proportion	Std. err.	Binomial exact	
				[95% conf. interval]	
	1,000	.05	.006892	.0373354	.0653905

With  $S = 1000$ , the 95% simulation interval is  $[0.037, 0.065]$ . With  $S = 10000$  simulations, this interval narrows to  $[0.046, 0.054]$ .

In general, tests rely on asymptotic theory, and we do not expect the true size to exactly equal the nominal size unless the sample size  $N$  is very large and the number of simulations  $S$  is very large. In this example, with 150 observations and only 1 regressor, the asymptotic theory performs well even though the model error is skewed.

### 11.7.3 Simulation using actual data

In the preceding example, the dataset was one artificially constructed from a completely specified DGP. What if one instead wants to base the simulation on data very similar to the dataset at hand.

One approach is to fit a regression model using actual data and then use this model to generate in each simulation round a new vector of dependent variables.

Building on the preceding example, we might with real data do OLS regression of  $y_i$  on  $\mathbf{x}_i$  giving estimates  $\hat{\beta}$  and  $\hat{\sigma}^2$ . Then, in the  $s$ th simulation generate  $y_{is} = \mathbf{x}'_i \hat{\beta} + u_{is}$ ,  $i = 1, \dots, N$ , where  $u_{is}$  is a draw from a right-skewed distribution with mean 0 and variance  $\hat{\sigma}^2$ . OLS estimation then gives the  $s$ th estimate  $\hat{\beta}^{(s)}$ . One could then, for example, test at 5% in each

simulation the hypothesis that  $\beta_j$ , the  $j$ th component of  $\beta$ , equals the DGP value of  $\beta_j$ , which is just the original OLS estimate  $\hat{\beta}_j$ . The hypothesis should be rejected in 5% of simulations. A variation could generate data with  $\beta_j = 0$  and test whether  $\beta_j = 0$ .

A second approach called a placebo approach does the following. Let  $z_i$  denote the key regressor of interest and  $\mathbf{x}_i$  denote the other control variables. In the  $s$ th simulation, generate  $z_{is}, i = 1, \dots, N$ , where  $z_{is}$  has properties similar to those of the original dataset  $z_i$ . Then, in each simulation, perform OLS regression of  $y_i$  on  $z_{is}$  and  $\mathbf{x}_i$ . Because  $z_{is}$  was generated independently of the data on  $y_i$  and  $\mathbf{x}_i$ , we expect no relationship so that tests at level 0.05 of the hypothesis that the coefficient of  $z_i$  equals 0 should reject 5% of the time.

For simplicity, we considered OLS estimation with independent data, but the methods can clearly be adapted to other estimators and to correlated data such as clustered data.

#### 11.7.4 Test power

A second error in testing, called a type II error, is to fail to reject  $H_0$  when we should reject  $H_0$ . The power of a test is one minus the probability of making this error. Thus,

$$\text{Power} = \Pr(\text{Reject } H_0 | H_0 \text{ false})$$

Ideally, test size is minimized and test power is maximized, but there is a tradeoff with smaller size leading to lower power. The standard procedure is to set the size at a level such as 0.05 and then use the test procedure that is known from theory or simulations to have the highest power.

The power of a test is not reported because it needs to be evaluated at a specific  $H_a$  value, and the alternative hypothesis  $H_a$  defines a range of values for  $\beta$  rather than one single value.

We compute the power of our test of  $\beta_2 = \beta_2^{Ha}$  against  $H_a: \beta_2 = \beta_2^{Ha}$ , where  $\beta_2^{Ha}$  takes on a range of values. We do so by first writing a program that determines the power for a given value  $\beta_2^{Ha}$  and then calling this program many times to evaluate at the many values of  $\beta_2^{Ha}$ .

The program is essentially the same as that used to determine test size, except that the command generating  $y$  becomes `generate y = 1 + b2Ha*x + rchi2(1)-1`. We allow more flexibility by allowing the user to pass the number of simulations, sample size,  $H_0$  value of  $\beta_2$ ,  $H_a$  value of  $\beta_2$ , and nominal test size ( $\alpha$ ) as the arguments, respectively, `num sims`, `num obs`, `b2H0`, `b2Ha`, and `nominal size`. The r-class program returns the computed power of the test as the scalar `p`. We have

```
* Program to compute power of test given specified H0 and Ha values of b2
program mypower, rclass
    version 17
    args num sims num obs b2H0 b2Ha nominal size
                                // Setup before simulation loops
    drop _all
    set seed 10101
    postfile sim pvalues using power, replace
                                // Simulation loop
    forvalues i = 1/`num sims' {
        drop _all
        quietly set obs `num obs'
        quietly generate double x = rchi2(1)
        quietly generate y = 1 + `b2Ha'*x + rchi2(1)-1
        quietly regress y x
        quietly test x = `b2H0'
        scalar p = r(p)
        post sim (p)
    }
    postclose sim
    use power, clear
                                // Determine the size or power
    quietly count if pvalues < `nominal size'
    return scalar power=r(N)/`num sims'
end
```

This program can also be used to find the size of the test of  $H_0: \beta_2 = 2$  by setting  $\beta_2^{Ha} = 2$ . The following command obtains the size using 1,000 simulations and a sample size of 150 for a test of the nominal size 0.05.

```

. * Size = power of test of b2H0=2 when b2Ha=2, S=1000, N=150, alpha=0.05
. mypower 1000 150 2.00 2.00 0.05
(file power.dta not found)
. display r(power) " is the test size"
.05 is the test size

```

The program `power` uses exactly the same coding as that given earlier for size computation; we have the same number of simulations and same sample size, and we get the same size result of 0.05.

To find the test power, we set  $\beta_2 = \beta_2^{Ha}$ , where  $\beta_2^{Ha}$  differs from the null hypothesis value. Here we set  $\beta_2^{Ha} = 2.2$ , which is approximately 2.4 standard errors away from the  $H_0$  value of 2.0 because, from section [5.6.1](#), the standard error of the slope coefficient is 0.084. We obtain

```

. * Power of test of b2H0=2 when b2Ha=2.2, S=1000, N=150, alpha=0.05
. mypower 1000 150 2.00 2.20 0.05
. display r(power) " is the test power"
.663 is the test power

```

Ideally, the probability of rejecting  $H_0: \beta_2 = 2.0$  when  $\beta_2 = 2.2$  is 1. In fact, it is only 0.663.

We next evaluate the power for a range of values of  $\beta_2^{Ha}$ , here from 1.60 to 2.40 in increments of 0.025. We use the `postfile` command, which was presented in section [5.3.4](#):

```

. * Power of test of H0:b2=2 against Ha:b2=1.6,1.625, ..., 2.4
. postfile simofsims b2Ha power using simresults, replace
(file simresults.dta not found)
. forvalues i = 0/33 {
2.     drop _all
3.     scalar b2Ha = 1.6 + 0.025*i'
4.     mypower 1000 150 2.00 b2Ha 0.05
5.     post simofsims (b2Ha) (r(power))
6. }
. postclose simofsims
. use simresults, clear
. summarize

```

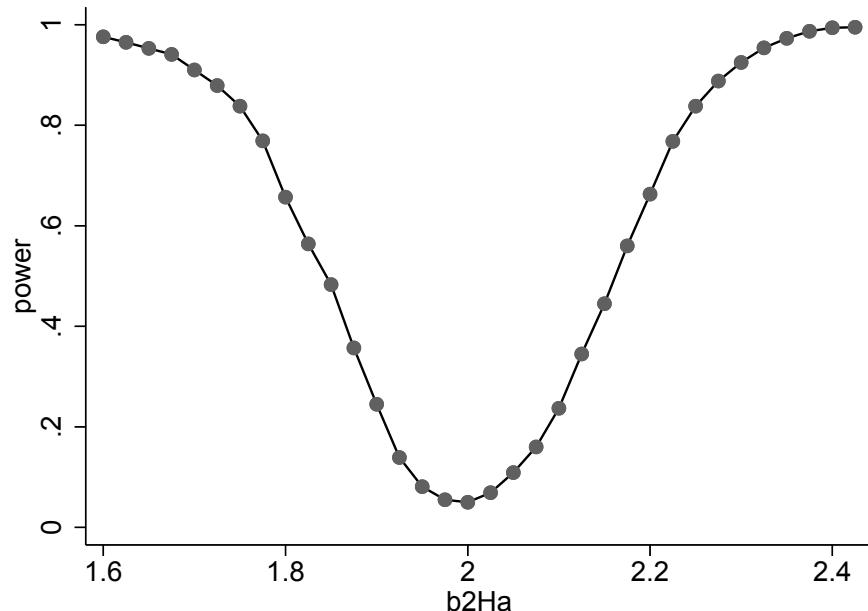
Variable	Obs	Mean	Std. dev.	Min	Max
b2Ha	34	2.0125	.2489562	1.6	2.425
power	34	.6109412	.3480345	.05	.995

The simplest way to see the relationship between power and  $\beta_2^{Ha}$  is to plot the power curve.

```

. * Plot the power curve
. twoway (connected power b2Ha), scale(1.2) plotregion(style(none))

```



**Figure 11.2.** Power curve for the test of  $H_0: \beta_2 = 2$  against  $H_a: \beta_2 \neq 2$  when  $\beta_2$  takes on the values  $\beta_2^{Ha} = 1.6, \dots, 2.4$

under  $H_a$  and  $N = 150$  and  $S = 1000$

As you can see in figure 11.2, power is minimized at  $\beta_2^{Ha} = \beta_2^{H0} = 2$ , and then the power size equals 0.05, as desired. As  $|\beta_2^{Ha} - \beta_2^{H0}|$  increases, the power goes to 1, but power does not exceed 0.9 until  $|\beta_2^{Ha} - \beta_2^{H0}| > 0.3$ .

The power curve can be made smoother by increasing the number of simulations or by smoothing the curve by, for example, using predictions from a regression of `power` on a quartic in `b2Ha`. Alternatively, by appropriately writing the code to compute power, one can obtain graphs and tables using options of Stata's `power` command; see [PSS-2] *power usermethod*.

### 11.7.5 Asymptotic test power

The asymptotic power of the Wald test can be obtained without resorting to simulation. We do this now for the square of the  $t$  test.

We consider  $W = \{(\hat{\beta}_2 - 2)/s_{\hat{\beta}_2}\}^2$ . Then  $W \xrightarrow{a} \chi^2(1)$  under  $H_0: \beta_2 = 2$ . It can be shown that under  $H_a: \beta_2 = \beta_2^{Ha}$ , the test statistic  $W \xrightarrow{a} \text{noncentral } \chi^2(1; \lambda)$ , where the noncentrality parameter  $\lambda = (\beta_2^{Ha} - 2)^2/\sigma_{\hat{\beta}_2}^2$ . [If  $\mathbf{y} \sim N(\boldsymbol{\delta}, \mathbf{I})$ , then  $(\mathbf{y} - \boldsymbol{\delta})'(\mathbf{y} - \boldsymbol{\delta}) \sim \chi^2(1)$  and  $\mathbf{y}'\mathbf{y} \sim \text{noncentral } \chi^2(h; \boldsymbol{\delta}'\boldsymbol{\delta})$ , where  $h = \dim(\mathbf{y})$ .]

We again consider the power of a test of  $\beta_2 = 2$  against  $\beta_2^{Ha} = 2.2$ . Then  $\lambda = (\beta_2^{Ha} - \beta_2^{H0})^2/\sigma_{\hat{\beta}_2}^2 = (0.2/0.084)^2 = 5.67$ , where we recall the earlier discussion that the DGP is such that  $\sigma_{\hat{\beta}_2} = 0.084$ . A  $\chi^2(1)$  test rejects at a level of  $\alpha = 0.05$  if  $W > 1.96^2 = 3.84$ . So the asymptotic test power equals  $\Pr\{W > 3.84 | W \xrightarrow{a} \text{noncentral } \chi^2(1; 5.67)\}$ . The `nchi2()` function gives the relevant c.d.f., and we use `1-nchi2()` to get the right tail. We have

```
. * Power of chi(1) test when noncentrality parameter lambda = 5.634
. display 1-nchi2(1,5.634,3.841)
.66048164
```

The asymptotic power of 0.660 is similar to the estimated power of 0.663 from the Monte Carlo example of the preceding subsection. This closeness

of asymptotic power to finite sample power is due to the relatively large sample size with just one regressor.

## 11.8 The power onemean command for multiple regression

Let  $\theta_0$  and  $\theta_a$  denote values of  $\theta$  under, respectively, the null and alternative hypotheses. Then the difference

$$\delta = (\theta_a - \theta_0) \quad (11.9)$$

is called the effect size. In econometrics applications, it is common for  $\theta_0 = 0$ .

Power calculations, done especially in design of a lab experiment or field experiment, consider three standard calculations for a test of size  $\alpha$ .

1. Calculate power ( $\pi$ ) for a given effect size ( $\delta$ ) and a given sample size ( $N$ ).
2. Calculate minimum effect size necessary to attain a given level of power with a given sample size.
3. Calculate the minimum sample size to attain a given power and desired effect.

The `power` commands provide such calculations for tests on means, proportions, correlations, variances, and difference in these quantities, in an i.i.d. setting. Section 24.3 presents in detail the `power twomeans` command for power calculations for a binary treatment. The `power` command also includes methods for analysis of variance, linear regression (including slope coefficient in bivariate regression with i.i.d. normal errors), contingency tables, and survival analysis. Additionally, one can add one's own method to the `power` command.

In this section, we show how to adapt the `power onemean` command to the more general case of linear or nonlinear multiple regression that yields an estimate  $\hat{\theta}$  that has an asymptotic normal distribution or, in the case of clustered data, has distribution approximated by a  $t$  distribution with degrees of freedom the number of clusters minus one.

### 11.8.1 The power onemean command

The `power onemean` command was designed for testing hypotheses on the population mean in an i.i.d. sample. To compute test power in this setting, use the command

```
power onemean m0 ma, n(numlist) [sd(numlist) options]
```

where *m0* and *ma* are values of the population mean under the null and alternative hypothesis, `sd()` is the standard deviation for a single observation, and `n()` is the number of observations.

To instead compute minimum effect size necessary to attain a given level of power for a particular test size, add the `power()` option and drop the *ma*.

```
power onemean m0, n(numlist) power(numlist) [sd(numlist) options]
```

And to compute minimum sample size necessary to attain a given level of power for a given effect size and test size, add the `power()` option and drop the `n()` option.

```
power onemean m0 ma [, power(numlist) sd(numlist) options]
```

The default is a two-sided test of size 0.05 using the  $t(N - 1)$  distribution. Options include `alpha()` to set the size of the test to a value different from 0.05, `knownsd` to give the power of a standard normal test, and `onesided` for a one-sided test. By providing a *numlist*, one can perform calculations over a range of values. The `table()` option can provide more compact output. The `graph()` option can plot results across a range of specified power, sample size, or effect size values.

Many of the other power commands have similar options, though most use the normal distribution rather than the  $t$  distribution. For several `power` commands, the option `rho()` allows for clustered data, where `rho()` is the intracluster correlation.

### 11.8.2 Power in a regression setting using the standard normal

For bivariate linear regression with i.i.d. normal the `power oneslope` command calculates power given, for example, the standard deviations of the regressor and errors. We instead analyze a much more general regression setting.

Consider a two-sided Wald test at level  $\alpha$  of  $H_0 : \theta = \theta_0$  for general estimator  $\hat{\theta} \stackrel{a}{\sim} N(\theta, s_{\hat{\theta}}^2)$ , where  $s_{\hat{\theta}}$  is the standard error of  $\hat{\theta}$ .

When standard normal critical values are used, the power ( $\pi$ ) of this test against  $H_a : \theta = \theta_a$  equals the probability that  $|(\hat{\theta} - \theta_0)/s_{\hat{\theta}}| > z_{\alpha/2}$  when  $\hat{\theta} \stackrel{a}{\sim} N(\theta_a, s_{\hat{\theta}}^2)$  because then  $(\hat{\theta} - \theta_a)/s_{\hat{\theta}} \sim N(0, 1)$ . It follows that

$$\beta = \Phi(\gamma - z_{\alpha/2}) + \Phi(-\gamma - z_{\alpha/2}), \quad \gamma = \delta/s_{\hat{\theta}} = (\theta_a - \theta_0)/s_{\hat{\theta}} \quad (11.10)$$

where by  $z_{\alpha/2}$  we mean the area in the right tail is  $z_{\alpha/2}$ .

#### Statistical significance test with size 0.05 and power 0.80

Before applying the `power` command, we note that a commonly used threshold in experimental design is that the effect size  $\delta = (\theta_a - \theta_0)$  should be large enough that a test of size 0.05 has power of at least 0.80. Solving the preceding equation for  $\gamma$  when  $\alpha = 0.05$  and  $\pi = 0.80$  yields  $\gamma = 2.8016$ .

For a two-sided test of statistical significance ( $\theta_0 = 0$ ), this norm means we should be able to detect an effect of magnitude  $\theta_a/s_{\hat{\theta}} = 2.8016$  or one that is at least 2.8016 standard errors from 0. By comparison, the usual norm in econometric studies is to ignore power and use a much lower threshold of  $z_{0.025} = 1.9600$ . The higher threshold of 2.8016 corresponds to determining a meaningful effect if the test  $p$ -value is less than 0.005 because  $z_{0.0025} = 2.8070$ . Thus, and also because of concerns about data mining and model mining, some leading researchers in some branches of applied statistics such as psychology are advocating that statistical significance tests of new discoveries should use a threshold of  $p < 0.005$  or, equivalently,  $|z| > 2.80$ .

Similar calculations for a one-sided test of statistical significance yields  $\gamma = \theta_a / s_{\hat{\theta}} = 2.4865$ , so statistical significance effects should use a threshold of  $p < 0.0065$  because  $z_{0.0065} = 2.4838$  or, equivalently,  $|z| > 2.4865$ .

#### Compute test power for given effect size and test size

The `power onemean` command, designed for statistical inference on the mean in an i.i.d. sample, can be adapted to the current more general case of inference on  $\theta$ . To do so, we use options `knownsd n(1) sd(s $\hat{\theta}$ )`. More generally, we can equivalently use options `n(w) sd(sqrt(w) * s $\hat{\theta}$ )` for any positive integer  $w$ ; it is simplest to use  $w = 1$ . Note that options `n()` and `sd()` require numbers or lists of numbers, not expressions.

We continue with the example of the preceding section. Then  $s_{\hat{\theta}} = 0.0843$ , and we perform a two-sided test of  $\theta_0 = 2$  and obtain power against  $\theta_a = 2.2$ . We obtain

```
. * Power of Wald normal test: 2 versus 2.2, size 0.05, s_thetahat=0.0843
. power onemean 2 2.2, knownsd alpha(0.05) n(1) sd(0.0843)
Estimated power for a one-sample mean test
z test
H0: m = m0  versus  Ha: m != m0
Study parameters:
    alpha =      0.0500
        N =          1
    delta =      2.3725
       m0 =      2.0000
       ma =      2.2000
        sd =      0.0843
Estimated power:
    power =      0.6600
```

We obtain asymptotic power of 0.660 as in the previous subsection.

Because we used options `n(1) sd(s $\hat{\theta}$ )`, the variable `delta` in the output measures the effect size in units of standard errors:  
 $(\theta_a - \theta_0) / s_{\hat{\theta}} = (2.2 - 2.0) / 0.0843 = 2.3725$ .

#### Compute minimum effect size for given test size and power

In this example, the power is still a long way from the desired 1.0. Just as a common choice of test size is 0.05, a common choice in the biostatistics literature for desired power is 0.80. Combining, we see the probability of a type 1 error is then 0.05, while the probability of a type 2 error is at most 0.20.

We wish to obtain the minimum effect size that would give power of 0.80, where effect size is given by  $(\theta_a - \theta_0)$ , for a test of size 0.05. The `power onemean` command can be used, adding the `power()` option and dropping the `ma` option. Continuing the preceding example, we have

```
. * Min effect size of normal test: 2 versus ?, size 0.05, power 0.80, s_t=.0843
. power onemean 2, knownsd alpha(0.05) power(0.80) n(1) sd(0.0843)
>     table(alpha power m0 ma sd N delta, formats(power "%7.3f"))
Performing iteration ...
Estimated target mean for a one-sample mean test
z test
H0: m = m0  versus  Ha: m != m0; ma > m0
```

alpha	power	m0	ma	sd	N	delta
.05	0.800	2	2.236	.0843	1	2.802

The minimum effect size is 2.802, so we need  $\hat{\theta}$  to be at least 2.802 standard errors from the null hypothesis value of 2.0, which requires  $\hat{\theta} \geq 2.236$ .

As already noted, the usual test for statistical significance at 5% has the less stringent requirement that  $\hat{\theta}$  be at least 1.960 standard errors from 2.0, or  $\hat{\theta} \geq 2.0 + 1.960 \times 0.0843 = 2.165$ .

#### Compute minimum sample size for desired effect size and given size and power

Now suppose we want to determine the sample size that achieves a given level of test size and power for a desired effect size.

Unlike the preceding examples, we need to make the additional assumption that an  $m$  times increase in sample size leads to the standard error of  $\hat{\theta}$  being  $1/m$  times as large. This may be reasonable when observations are independent; more difficult adaptation to correlated observations is given in a later subsection.

For power and effect size calculations, we could use options `n(w) sd(`  
 $\sqrt{w} \times s_{\hat{\theta}}$ ) for any integer  $w$ ; we used  $w = 1$  for simplicity. For minimum sample-size computations, we need to use  $w = N$ , where  $N$  is the sample size.

In the preceding examples, the standard error of 0.0843 is based on a sample size of 150. So we use option `sd(1.0325)` because

$\sqrt{N} \times s = \sqrt{150} \times 0.0843 = 1.0325$ . We additionally use the `table()` option to obtain more compact output.

```
. * Min sample size for normal test: 2 versus 2.2, size 0.05, power 0.80,
> independent
. di sqrt(150)*0.0843
1.0324599
. power onemean 2 2.2, knownsd alpha(0.05) power(0.80) sd(1.0325)
>     table(alpha power m0 ma sd N delta, formats(power "%7.3f"))
Performing iteration ...
Estimated sample size for a one-sample mean test
z test
H0: m = m0  versus  Ha: m != m0
```

alpha	power	m0	ma	sd	N	delta
.05	0.800	2	2.2	1.032	210	.1937

We need at least 210 observations.

#### Compute a power curve

The `power` command allows some of the arguments to be given as number lists that span a range of values.

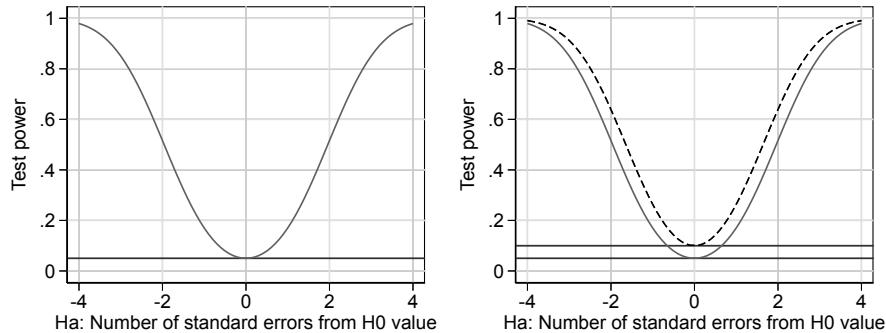
For example, suppose we want to plot a power curve of asymptotic power against the number of standard errors that the alternative hypothesis value is from the null hypothesis value. In that case, we set `sd` equal to 1 and the `m0` value to 0. To consider alternative values that are up to 4 standard errors from the  $H_0$  value, for example, give a range of values of `ma` from  $-4$  to  $4$ , here at intervals of  $0.1$ . The `graph()` option of the `power onemean` command is used to plot the power curve.

We obtain

```
. * General power curve for N(0,1) test at size = 0.05
. clear

. power onemean 0 (-4(0.1)4), knownsd sd(1) n(1) alpha(0.05)
> graph(legend(off) scale(1.2) title("") subtitle(""))
> note("") recast(line)
> ylabel(, grid angle(0)) yline(0.05) xlabel(, grid)
> xtitle("Ha: Number of standard errors from H0 value")
> ytitle("Test power")
```

The first panel of figure 11.3 plots the power curve. As expected, the power curve for a test at level 0.05 takes a minimum value of 0.05 at the null hypothesis value of  $\theta$ . Power is only 0.50 at the 5% test critical value of 1.96. Power is 0.80 when the alternative value is approximately 2.8 standard errors from the null hypothesis value.



**Figure 11.3.** Power curves at test size 0.05 and at sizes 0.05 and 0.10

The second panel of figure 11.3 plots power curves at two different test sizes. The test at size  $\alpha = 0.10$ , given by the dashed line, has uniformly higher power than the test at size  $\alpha = 0.05$ ; increasing the probability of a type 1 error decreases the probability of a type 2 error. The critical value for a test of size  $\alpha = 0.10$  is  $z_{.050} = 1.645$ , and the power at 1.645 standard deviations equals approximately 0.50, as expected.

### 11.8.3 Power in a regression setting using the t distribution

We now consider the same two-sided Wald test at level  $\alpha$  of  $H_0 : \theta = \theta_0$  for general estimator  $\hat{\theta} \stackrel{a}{\sim} N(\theta, s_{\hat{\theta}}^2)$  but now use critical values from the students  $t$  distribution with  $v$  degrees of freedom, where typically  $v = N - K$  when observations are independent and  $v = G - 1$  when observations are clustered with  $G$  clusters.

We compute the power of this test against  $H_a : \theta = \theta_a$ . The power is the probability that  $|(\hat{\theta} - \theta_0)/s_{\hat{\theta}}| > t_{v;\alpha/2}$  when the distribution of  $\hat{\theta}$  is centered on  $\theta_a$  rather than on  $\theta_0$ . Translating the centering of a normally distributed random variable leads again to a normally distributed variable. By contrast, translating the centering of a  $t$ -distributed random variable leads to a random variable with the noncentral  $t$  distribution. The noncentrality parameter is the amount of the translation, here  $\gamma = (\theta_a - \theta_0)/s_{\hat{\theta}}$ . Then,

$$\pi = 1 - T_{v,\gamma}(t_{v,\alpha/2}) + T_{v,\gamma}(-t_{v,\alpha/2}), \quad \gamma = (\theta_a - \theta_0)/s_{\hat{\theta}} \quad (11.11)$$

where  $T_{v,\gamma}$  is the c.d.f. of the noncentral  $t$  distribution with  $v$  degrees of freedom and noncentrality parameter  $\gamma$ , and by  $t_{v,\alpha/2}$ , we mean the area in the right tail of the central  $t(v)$  distribution is  $\alpha/2$ .

#### Compute test power for given effect size and test size

The `power.onemean` command, designed for inference on the mean using the  $t(n - 1)$  distribution, is easily adapted to this more general setting for power and effect size calculations with constant sample size. We use options `n` ( $\nu + 1$ ) `sd` ( $\sqrt{\nu + 1} \times s_{\hat{\theta}}$ ).

For example, suppose our current example came from clustered data with 12 clusters, so we base inference on the  $t(11)$  distribution. Then, we use options `n(12)` and `sd(0.29202)` because  $\nu + 1 = 12$  and  $\sqrt{12} \times 0.0843 = 0.29202$ . We obtain

```

. * Power of Wald t(11) test: 2 versus 2.2, size 0.05, s_thetahat=0.0843
. di sqrt(12)*0.0843
.29202377
. power onemean 2 2.2, alpha(0.05) n(12) sd(0.29202)
> table(alpha power m0 ma sd N delta, formats(power "%7.3f"))
Estimated power for a one-sample mean test
t test
H0: m = m0  versus  Ha: m != m0

```

alpha	power	m0	ma	sd	N	delta
.05	0.580	2	2.2	.292	12	.6849

The power for the  $t$  test is 0.580, lower than 0.660 for the standard normal test. Such differences become greater as the degrees of freedom fall. Note that the result `delta` no longer measures the effect size in units of standard errors. This becomes especially relevant in clustered settings considered below.

#### Compute minimum effect size for given test power and size

We next obtain the minimum effect size that would give power of 0.80, where effect size is given by  $(\theta_a - \theta_0)$ , for a test of size 0.05. Continuing the preceding  $t(11)$  example, we again use options `n(12)` and `sd(0.29202)` because  $\nu + 1 = 12$  and  $\sqrt{12} \times 0.0843 = 0.29202$ . We obtain

```

. * Min effect size of t(11) test: 2 versus ?, size 0.05, power 0.80, s_t=.29202
. power onemean 2, alpha(0.05) power(0.80) n(12) sd(0.29202)
> table(alpha power m0 ma sd N delta, formats(power "%7.3f"))
Performing iteration ...
Estimated target mean for a one-sample mean test
t test
H0: m = m0  versus  Ha: m != m0; ma > m0

```

alpha	power	m0	ma	sd	N	delta
.05	0.800	2	2.26	.292	12	.8887

The minimum value of  $\theta_a$  is 2.260, so the minimum effect size is  $(\theta_a - \theta_0) = (2.260 - 2.0) = 0.260$ , compared with 0.236 using the standard normal. Using the finite sample adjustment, we require stronger evidence than if using asymptotic results. Equivalently, the reported  $\delta$  equals the

minimum effect size multiplied by  $\sqrt{\nu + 1}$ , so the minimum effect size equals  $0.8887/\sqrt{12} = 0.260$ . We need  $\hat{\theta}$  to be at least  $0.260/0.0843 = 3.08$  standard errors from the null hypothesis value of 2.0.

#### Compute minimum sample size for desired effect size and given size and power

When  $t$  tests are used, extending the `power onemean` command to regression estimate  $\hat{\theta}$  is complicated.

The simplest approach may be to instead directly compute the power for varying sample sizes and associated values of  $s_{\hat{\theta}}$  and degrees of freedom.

Thus, consider the example with  $\alpha = 0.05$ ,  $s_{\hat{\theta}} = 0.0843$ ,  $\hat{\theta}$  with  $s_{\hat{\theta}} = 0.843$ , and inference based on the  $t(11)$  distribution. Then, using [\(11.11\)](#), we see the test power is

```
. * Power for t(11) with specified test size and effect size
. scalar df = 11
. scalar se = 0.0843
. scalar effectsize = 0.2
. scalar ncp = effectsize/se
. scalar size = 0.05
. scalar tcrit = invtail(df, size/2)
. display "power = " 1 - nt(df, ncp, tcrit) + - nt(df, ncp, -tcrit)
power = .5804467
```

The power is 0.580, as was obtained previously applying the `power onemean` command to the same example.

Now suppose we double the sample size and assume that this leads to  $\text{Var}(\hat{\theta})$  halving, so the initial  $s_{\hat{\theta}}$  is divided by  $\sqrt{2}$ . And suppose that the degrees of freedom increase to 23, which would be the case if the number of clusters increased from 12 to 24, using  $G - 1$  degrees of freedom.

```

. * Power with df increase from 11 to 23 (for example, double the number of
> clusters)
. scalar df = 23
. scalar se = 0.0843/sqrt(2)
. scalar effectsize = 0.2
. scalar ncp = effectsize/se
. scalar size = 0.05
. scalar tcrit = invtail(df, size/2)
. display "power = " 1 - nt(df, ncp, tcrit) + - nt(df, ncp, -tcrit)
power = .89475003

```

The power has increased to 0.895. With trial and error, we could find the sample size so that power equals 0.80.

This example is of most importance in the clustered case, where degrees of freedom are often small.

#### 11.8.4 Power for clustered data

The `power onemean, cluster` command permits clustered data, assuming equicorrelation within cluster. The `rho()` option specifies the intracluster correlation, the `k()` option specifies the number of clusters, the `m()` option specifies the cluster size, and the `cvccluster()` option specifies the coefficient of variation for cluster sizes should clusters not be equal sized.

These options are not suited to the current focus of extension to a slope coefficient in multiple regression, so they are not presented here. The `power twomeans` command with clustering options is presented in section 24.3 for an randomized control trial with a binary treatment.

Instead, we continue with the preceding approach, using the *t* distribution rather than the normal distribution to better approximate distributions when there are few clusters.

With clustered data, especially with few clusters, one should use the  $t(G - 1)$  distribution, where  $G$  is the number of clusters, rather than the standard normal distribution.

#### Compute test power and compute effect size

The preceding power and minimum effect size analysis for  $t$  distribution carries through, with  $v = G - 1$  degrees of freedom.

#### Compute the minimum number of clusters

Suppose we wish to compute the minimum number of clusters necessary to achieve a desired test size, power, and effect size, based on regression on an existing dataset with  $G_1$  cluster increases and standard error  $s_{\hat{\theta}}$ . If we assume that additional clusters are of similar size and informational content to the existing clusters, then adding  $G_2$  clusters leads to a new standard error equal to  $s_{\hat{\theta}}/\sqrt{(G_1 + G_2)/G_1}$ , and degrees of freedom increase to  $G_1 + G_2 - 1$ .

The power can be computed as in section [11.8.3](#), and a desired power attained by trial-and-error variation in  $G_2$ .

#### Compute sample size with fixed number of clusters

More challenging is the common situation where the number of clusters is given and we wish to find how many observations to include in each cluster. The preceding analysis no longer applies because increasing the number of observations in each cluster leads to smaller efficiency gains than usual because of within-cluster correlation.

For OLS one way to proceed is to use the variance inflation factor introduced in section [3.4.6](#). Using this, increasing the average number of observations per cluster from  $M_1$  to  $M_2$  leaves the degrees of freedom remain unchanged while the standard error decreases by the multiple  $\sqrt{1 + \rho_x \rho_u(M_1 - 1)}/\sqrt{1 + \rho_x \rho_u(M_2 - 1)}$ . Again the power can be computed as in section [11.8.3](#), and a desired power attained by trial and error variation in  $G_2$ .

### 11.8.5 Confidence intervals with desired precision

The `ciwidth` command is the confidence-interval analog of the `power` command.

As in section [11.8.2](#), consider an asymptotically normal distributed estimator  $\hat{\theta}$  with standard error of 0.0843 based on a sample size of 150, and assume that observations are independent. We use the `ciwidth onemean` command to obtain the width of a confidence interval given a specified confidence level and sample size and to obtain the minimum sample size needed to obtain a confidence interval of specified width and confidence level. To do so, we use option `sd(1.0325)` because the contribution of a single observation is  $\sqrt{N} \times s = \sqrt{150} \times 0.0843 = 1.0325$ .

For a sample size of 300, a 95% confidence interval will have width 0.2337, where the width is the distance between the lower and upper values of the confidence interval.

```
. * Confidence interval width given estimator precision and sample size
. ciwidth onemean, knownsd level(95) sd(1.0325) n(300)
>     table(level N sd width, formats(level "%7.3f"))

Estimated width for a one-mean CI
Normal two-sided CI
```

level	N	sd	width
95.000	300	1.032	.2337

And to obtain a 95% confidence interval of width 0.2, we need at least 410 observations.

```
. * Minimum sample size for confidence interval of given width
. ciwidth onemean, knownsd level(95) sd(1.0325) width(0.2)
>     table(level N sd width, formats(level "%7.3f"))

Estimated sample size for a one-mean CI
Normal two-sided CI
```

level	N	sd	width
95.000	410	1.032	.2

These calculations are based on normal distribution critical values and implicitly assume that the estimator precision is exactly estimated. When  $t$  distribution critical values are used, the `ciwidth onemean` command requires use of the `probwidth()` option to additionally allow for the randomness in the estimator precision.

## 11.9 Specification tests

The Wald, LR, and LM tests are often used for specification testing, particularly of inclusion or exclusion of regressors. In this section, we consider other specification testing methods that differ in that they do not work by directly testing restrictions on parameters. Instead, they test whether moment restrictions implied by the model, or other model properties, are satisfied. It is helpful to first read sections [13.3–13.4](#).

### 11.9.1 Moment-based tests

A moment-based test, or  $m$  test, is one of moment conditions imposed by a model but not used in estimation. Specifically,

$$H_0: E\{\mathbf{m}(y_i, \mathbf{x}_i, \boldsymbol{\theta})\} = \mathbf{0} \quad (11.12)$$

where  $\mathbf{m}(\cdot)$  is an  $h \times 1$  vector. Several examples follow below. The test statistic is based on whether the sample analogue of this condition is satisfied, that is, whether  $\widehat{\mathbf{m}}(\widehat{\boldsymbol{\theta}}) = \sum_{i=1}^N \mathbf{m}(y_i, \mathbf{x}_i, \widehat{\boldsymbol{\theta}}) = \mathbf{0}$ . This statistic is asymptotically normal because  $\widehat{\boldsymbol{\theta}}$  is, and taking the quadratic form, we obtain a chi-squared statistic. The  $m$ -test statistic is then

$$M = \widehat{\mathbf{m}}(\widehat{\boldsymbol{\theta}})' \left[ \widehat{V}\{\widehat{\mathbf{m}}(\widehat{\boldsymbol{\theta}})\} \right]^{-1} \widehat{\mathbf{m}}(\widehat{\boldsymbol{\theta}}) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0$$

As usual, we reject at the  $\alpha$  level if  $p = \Pr\{\chi^2(h) > W\} < \alpha$ .

Obtaining  $\widehat{V}\{\widehat{\mathbf{m}}(\widehat{\boldsymbol{\theta}})\}$  can be difficult. Often, this test is used after ML estimation because likelihood-based models impose many conditions that can be used as the basis for an  $m$  test. Then the auxiliary regression for the LM test (see section [11.5.3](#)) can be generalized. We compute  $M$  as  $N$  times the uncentered  $R^2$  from the auxiliary regression of 1 on  $\mathbf{m}(y_i, \mathbf{x}_i, \widehat{\boldsymbol{\theta}})$  and

$s_i(\hat{\theta})$ , where  $s_i(\theta) = \partial \ln f(y_i | \mathbf{x}_i, \theta) / \partial \theta$ . In finite samples, the test statistic has a size that can differ significantly from the nominal size, but this can be rectified by using a bootstrap with asymptotic refinement.

An example of this auxiliary regression, used to test moment conditions implied by the tobit model, is given in section 19.4.

### 11.9.2 Information matrix test

For a fully parametric model, the expected value of the outer product of the first derivatives of  $\ln L(\theta)$  equals the negative expected value of the second derivatives. This property, called the IM equality, enables the variance matrix of the MLE to simplify from the general sandwich form  $\mathbf{A}^{-1}\mathbf{B}\mathbf{A}^{-1}$  to the simpler form  $-\mathbf{A}^{-1}$ ; see section [13.4.4](#).

The IM test is a test of whether the IM equality holds. It is a special case of [\(11.12\)](#) with  $\mathbf{m}(y_i, \mathbf{x}_i, \theta)$  equal to the unique elements in  $s_i(\theta)s_i(\theta)' + \partial s_i(\theta) / \partial \theta$ . For the linear model under normality, the IM test is performed by using the `estat imtest` command after `regress`; see section [3.7](#) for an example.

### 11.9.3 Chi-squared goodness-of-fit test

A simple test of goodness of fit is the following. Discrete variable  $y$  takes on the values 1, 2, 3, 4, and 5, and we compare the fraction of sample values  $y$  that take on each value with the corresponding predicted probability from a fitted parametric regression model. The idea extends easily to partitioning on the basis of regressors as well as  $y$  and to continuous regressor  $y$ , where we replace a discrete value with a range of values.

Stata implements a goodness-of-fit test by using the `estat gof` command following `logit`, `logistic`, `probit`, and `poisson`. An example of `estat gof` following `logit` regression is given in section 17.5. A weakness of this command is that it treats estimated coefficients as known, ignoring estimation error. The goodness-of-fit test can instead be set up as an  $m$  test that provides additional control for estimation error; see [Andrews \(1988\)](#)

and [Cameron and Trivedi \(2005\)](#), 266–271). For count data, the community-contributed `chi2gof` command of [Manjón and Martínez \(2014\)](#) performs the correct chi-squared goodness-of-fit test.

### 11.9.4 Overidentifying restrictions test

In the generalized method of moments (GMM) estimation framework of section 16.8, moment conditions  $E\{\mathbf{h}(y_i, \mathbf{x}_i, \boldsymbol{\theta})\} = \mathbf{0}$  are used as the basis for estimation. In a just-identified model, the GMM estimator solves the sample analog  $\sum_{i=1}^N \mathbf{h}(y_i, \mathbf{x}_i, \hat{\boldsymbol{\theta}}) = \mathbf{0}$ . In an overidentified model, these conditions no longer hold exactly, and an overidentifying restrictions test is based on the closeness of  $\sum_{i=1}^N \mathbf{h}(y_i, \mathbf{x}_i, \hat{\boldsymbol{\theta}})$  to  $\mathbf{0}$ , where  $\hat{\boldsymbol{\theta}}$  is the optimal GMM estimator. The test is chi-squared distributed with degrees of freedom equal to the number of overidentifying restrictions.

This test is most often used in overidentified IV models, though it can be applied to any overidentified model. It is performed in Stata with the `estat overid` command after `ivregress gmm`; see section [7.4.8](#) for an example.

### 11.9.5 Hausman test

The Hausman test compares two estimators where one is consistent under both  $H_0$  and  $H_a$  while the other is consistent under  $H_0$  only. If the two estimators are dissimilar, then  $H_0$  is rejected. An example is to test whether a single regressor is endogenous by comparing two-stage least-squares and OLS estimates.

We want to test  $H_0 : \text{plim}(\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}}) = \mathbf{0}$ . Under standard assumptions, each estimator is asymptotically normal and so is their difference. Taking the usual quadratic form, we obtain

$$H = (\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}})' \left\{ \hat{V} (\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}}) \right\}^{-1} (\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}}) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0$$

The `hausman` command, available after many estimation commands, implements this test under the strong assumption that  $\widehat{\boldsymbol{\theta}}$  is a fully efficient estimator. Then it can be shown that  $\widehat{V}(\widehat{\boldsymbol{\theta}} - \widetilde{\boldsymbol{\theta}}) = \widehat{V}(\widetilde{\boldsymbol{\theta}}) - \widehat{V}(\widehat{\boldsymbol{\theta}})$ . In some common settings, the Hausman test can be more simply performed with a test of the significance of a subset of variables in an auxiliary regression. Both variants are demonstrated in section [8.8.5](#).

The standard microeconomics approach of using robust estimates of the VCE implicitly presumes that estimators are not efficient. Then the preceding test is incorrect. One solution is to use a bootstrapped version of the Hausman test; see section [12.4.6](#). A second approach is to test the statistical significance in the appropriate auxiliary regression by using robust standard errors; see sections [7.4.6](#) and [8.8.5](#) for examples.

### 11.9.6 Other tests

The preceding discussion only scratches the surface of specification testing. Many model-specific tests are given in model-specific reference books such as [Baltagi \(2021\)](#) for panel data, [Hosmer, Lemeshow, and Sturdivant \(2013\)](#) for binary data, and [Cameron and Trivedi \(2013\)](#) for count data. Some of these tests are given in estimation command output or through postestimation commands, usually as an `estat` command, but many are not.

## 11.10 Permutation tests and randomization tests

A permutation test provides a test of exact size by the following method. We consider settings where the data are exchangeable under the null hypothesis. For example, in a two-sample difference-in-means test, we assume that the two samples are from the same distribution under the null hypothesis of no difference in means.

First, compute the test statistic of interest using the original sample. Second, recompute this test statistic for every permutation of the data. Third, calculate the  $p$ -value as the fraction of times the permuted test statistics equal or exceed the value of the original sample test statistic. The test is a nonparametric test that does not require distributional assumptions, aside from the strong assumption that the data are exchangeable under the null hypothesis. Fisher proposed this exact test for contingency tables. It is well suited to randomized control trials.

In practice, with a reasonable number of observations, there are many permutations of the original data. A randomized permutation test randomly selects some of these permutations.

As an example, consider testing the difference in two means, which can be implemented by OLS regression of the outcome on an intercept and an indicator variable. The following example generates 100 observations, of which 47 have  $d = 1$ , with a weak relationship between  $y$  and  $d$ .

```
. * Permutation test example - DGP and usual test
. clear
. set obs 100
Number of observations (_N) was 0, now 100.
. set seed 10101
. gen d = runiform() > 0.5
. gen y = rnormal(1,1) + 0.2*d
. sum y d
```

Variable	Obs	Mean	Std. dev.	Min	Max
y	100	1.166272	.935289	-1.497009	2.873763
d	100	.47	.5016136	0	1

```
. regress y d, noheader
```

y	Coefficient	Std. err.	t	P> t	[95% conf. interval]
d	.2217631	.1870122	1.19	0.239	-.1493565 .5928827
_cons	1.062043	.1282091	8.28	0.000	.8076161 1.31647

The usual  $t$  test for statistical significance of  $d$  has  $p = 0.239$ .

Now we consider a permutation test where the 100 observations on  $y$  are randomly allocated (without replacement) to the unchanging 100 observations on  $d$ . In this example, this process is done 1,000 times leading to 1,000  $t$  statistics of  $H_0 : \beta_d = 0$ .

```
. * Permutation test
. capture program drop pols
. program pols, rclass
  1.      version 17
  2.      args y d
  3.      regress `y' `d'
  4.      return scalar beta = _b[d]
  5.      return scalar t = _b[d]/_se[d]
  6. end
. permute y t=r(t) beta=r(beta), nowarn nodots seed(10101) reps(1000): pols y d
Monte Carlo permutation results                               Number of observations = 100
Permutation variable: y                                     Number of permutations = 1,000
  Command: pols y d
            t: r(t)
            beta: r(beta)
```

T	T(obs)	Test	c	n	p	Monte Carlo error	
						SE(p)	[95% CI(p)]
t	1.185822	lower	879	1000	.8790	.0103	.8572 .8986
		upper	121	1000	.1210	.0103	.1014 .1428
		two-sided			.2420	.0135	.2155 .2685
beta	.2217631	lower	879	1000	.8790	.0103	.8572 .8986
		upper	121	1000	.1210	.0103	.1014 .1428
		two-sided			.2420	.0135	.2155 .2685

Notes: For lower one-sided test,  $c = \#\{T \leq T(\text{obs})\}$  and  $p = p_{\text{lower}} = c/n$ .

For upper one-sided test,  $c = \#\{T \geq T(\text{obs})\}$  and  $p = p_{\text{upper}} = c/n$ .

For two-sided test,  $p = 2*\min(p_{\text{lower}}, p_{\text{upper}})$ ; SE and CI approximate.

The permutation test has a two-sided  $p$ -value of 0.242, close to the 0.239 obtained from the original sample regression.

[Imbens and Rubin \(2015\)](#), chap. 5) provide a detailed presentation of Fisher's exact tests. [Young \(2019\)](#) contrasts randomization tests with conventional tests for many published randomized trials, and includes the typical complication of clustered trials.

Extending permutation tests from regression with a single regressor to multiple regression is difficult and is a topic of current research. It is possible if the regressor of interest is uncorrelated with other regressors, as is the case if the regressor is a randomly assigned treatment.

Randomization tests provide a general method to control test size in finite samples, under the assumption that the distribution of the data is invariant under a group of transformations under the null hypothesis, a property called symmetry. [Canay, Romano, and Shaikh \(2017\)](#) extend this theory to settings where the limiting distribution of a function of the data is symmetric under the null hypothesis. [Cai et al. \(Forthcoming\)](#) detail implementation. This enables finite-sample inference with few clusters and many observations per cluster.

## 11.11 Additional resources

The Stata documentation [FN] **Statistical functions**, `help density` functions, or `help` functions describes the functions to compute  $p$ -values and critical values for various distributions. For testing, see the relevant entries for the commands discussed in this chapter: [R] **test**, [R] **testnl**, [R] **lincom**, [R] **nlcom**, [R] **lrtest**, [R] **hausman**, [R] **regress postestimation** (for `estat imtest`), and [R] **estat**. For power analysis, see [PSS] *Stata Power, Precision, and Sample-Size Reference Manual*.

Much of the material in this chapter is covered in [Cameron and Trivedi \(2005\)](#), chap. 7 and 8) and [Hansen \(2022a\)](#), chap. 9) and various chapters of [Greene \(2018\)](#) and [Wooldridge \(2010\)](#). Multiple testing is a more recent topic.

## 11.12 Exercises

- For a  $\chi^2(h)$  random variable, the density is given by  

$$f(y) = \{y^{(h/2)-1} \exp(-y/2)\}/\{2^{h/2}\Gamma(h/2)\}$$
, where  $\Gamma(\cdot)$  is the gamma function and  $\Gamma(h/2)$  can be obtained as `exp(1ngamma(h/2))`. Plot this density for  $h = 5$  and  $y \leq 25$ .
- Use Stata commands to find the appropriate  $p$ -values for  $t(100)$ ,  $F(1, 100)$ ,  $Z$ , and  $\chi^2(1)$  distributions at  $y = 2.5$ . For the same distributions, find the critical values for tests at the 0.01 level.
- Consider the Poisson example in section [11.3.3](#), with a robust estimate of the VCE. Use the `test` or `testnl` commands to test the following hypotheses: 1)  $H_0: \beta_{\text{female}} - 100 \times \beta_{\text{income}} = 0.5$ ; 2)  $H_0: \beta_{\text{female}} = 0$ ; 3) test the previous two hypotheses jointly with the `mtest` option; 4)  $H_0: \beta_{\text{female}}^2 = 0$ ; and 5)  $H_0: \beta_{\text{female}}^{\beta_{\text{income}}} = 1$ . Are you surprised that the second and fourth tests lead to different Wald test statistics?
- Consider the test of  $H_0: \beta_{\text{female}}/\beta_{\text{private}} - 1 = 0$ , given in section [11.3.6](#). It can be shown that, given that  $\beta$  has the entries  $\beta_{\text{private}}$ ,  $\beta_{\text{chronic}}$ ,  $\beta_{\text{female}}$ ,  $\beta_{\text{income}}$ , and  $\beta_{\text{cons}}$ , then  $\widehat{\mathbf{R}}$  defined in [\(11.5\)](#) is given by

$$\widehat{\mathbf{R}} = \begin{bmatrix} -\frac{\widehat{\beta}_{\text{female}}}{(\widehat{\beta}_{\text{private}})^2} & 0 & \frac{1}{\widehat{\beta}_{\text{private}}} & 0 & 0 \end{bmatrix}$$

Manually calculate the Wald test statistic defined in [\(11.5\)](#) by adapting the code at the end of section [11.3.3](#).

- The claim is made that the effect of private insurance on doctor visits is less than that of having a chronic condition, that is, that  $\beta_{\text{private}} - \beta_{\text{chronic}} < 0$ . Test this claim at the 0.05 level. Obtain 95% and 99% confidence intervals for  $\beta_{\text{private}} - \beta_{\text{chronic}}$ .
- Consider the negative binomial example in section [11.4.1](#), where we test  $H_0: \alpha = 0$ . Use the output from the `nbreg` command to compute the Wald test statistic, and compare this with the LR test statistic given in the output. Next, calculate this Wald test statistic using the `testnl` command as follows: Fit the model by using `nbreg`, and then type

`estat vce`. You will see that the estimate `lnalpha` is saved in the equation `lnalpha` with the name `_cons`. We want to test  $\alpha = 0$ , in which case  $\exp(\ln \alpha) = 0$ . Give the command `testnl exp([lnalpha]_cons)=0`. Compare the results with your earlier results.

7. Consider the Poisson example of section [11.3.3](#). The parametric Poisson model imposes the restriction that  $\text{Var}(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ . An alternative model is that  $\text{Var}(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta}) + \alpha \times \exp(\mathbf{x}'\boldsymbol{\beta})^2$ . A test of overdispersion is a test of  $H_0: \alpha \leq 0$  against  $H_a: \alpha > 0$ . The LM test statistic can be computed as the  $t$  test that  $\alpha = 0$  in the auxiliary OLS regression of  $\{(y_i - \hat{\mu}_i)^2 - \hat{\mu}_i\}/\hat{\mu}_i$  on  $\hat{\mu}_i$  (no intercept), where  $\hat{\mu}_i = \exp(\mathbf{x}'\hat{\boldsymbol{\beta}})$ . Perform this test.
8. Consider the DGP  $y = 0 + \beta_2 x + \varepsilon$ ,  $x \sim N(0, 1)$ ,  $N = 36$ . For this DGP, what do you expect  $\text{Var}(\hat{\beta}_2)$  to equal? Consider a test of  $H_0: \beta_2 = 0$  at the 0.05 level. By simulation, find the size of the test and the power when  $\beta_2 = 0.25$ .
9. Consider the same DGP as in the previous question, but adapt it to a probit model by defining  $y^* = 0 + \beta_2 x + \varepsilon$ , and define  $y = 1$  if  $y^* > 0$  and  $y = 0$  otherwise. Consider a test of  $H_0: \beta_2 = 0$  at the 0.05 level. By simulation, find the size of the test and the power when  $\beta_2 = 0.25$ .



# **Chapter 12**

## **Bootstrap methods**

## 12.1 Introduction

Bootstrap methods enable statistical inference by resampling the data. The most common use of the bootstrap in microeconomics applications is to provide standard error estimates when analytical expressions are quite complicated. These standard errors are then used to form confidence intervals and test statistics.

Additionally, a more complicated bootstrap with asymptotic refinement can provide tests with an actual size closer to the nominal size and confidence intervals with an actual coverage rate closer to the nominal coverage rate, compared with the standard inferential methods presented in the preceding chapter. The leading microeconomics application of a bootstrap with asymptotic refinement is the wild cluster bootstrap.

## 12.2 Bootstrap methods

A bootstrap provides a way to perform statistical inference by resampling from the sample. The statistic being studied is usually a standard error, a confidence interval, or a test statistic.

### 12.2.1 Bootstrap estimate of standard error

As a leading example, consider calculating the standard error of an estimator  $\hat{\theta}$  when this is difficult to do using conventional methods.

Suppose 400 random samples from the population were available. Then we could obtain 400 different estimates of  $\hat{\theta}$  and let the standard error of  $\hat{\theta}$  be the standard deviation of these 400 estimates.

In practice, however, only one sample from the population is available. The bootstrap generates multiple samples by resampling from the current sample. Essentially, the observed sample is viewed as the population, and the bootstrap is a method to obtain multiple samples from this population. Given 400 bootstrap resamples, we obtain 400 estimates and then estimate the standard error of  $\hat{\theta}$  by the standard deviation of these 400 estimates.

Let  $\hat{\theta}_1^*, \dots, \hat{\theta}_B^*$  denote the estimates, where here  $B = 400$ . Then the bootstrap estimate of the variance of  $\hat{\theta}$  is

$$\hat{V}_{\text{Boot}}(\hat{\theta}) = \frac{1}{B-1} \sum_{b=1}^B \left( \hat{\theta}_b^* - \bar{\hat{\theta}}^* \right)^2 \quad (12.1)$$

where  $\bar{\hat{\theta}}^* = 1/B \sum_{b=1}^B \hat{\theta}_b^*$  is the average of the  $B$  bootstrap estimates.

The square root of  $\hat{V}_{\text{Boot}}(\hat{\theta})$ , denoted by  $\text{se}_{\text{Boot}}(\hat{\theta})$ , is called the bootstrap estimate of the standard error of  $\hat{\theta}$ . Some authors more correctly call this the bootstrap standard error of  $\hat{\theta}$ , or the bootstrap estimate of the

standard deviation of  $\hat{\theta}$ , because the term “standard error” means estimated standard deviation.

### 12.2.2 Bootstrap methods

The plural form, bootstrap methods, is used because there is no one single bootstrap. As already noted, the bootstrap can be used to obtain the distribution of many different statistics. There are several different ways to obtain bootstrap resamples. Even for a given statistic and bootstrap resampling method, there are different ways to proceed.

The simplest bootstraps implement standard asymptotic methods. The most common use of the bootstrap in microeconomics is standard error estimation. More complicated bootstraps implement more refined asymptotics.

### 12.2.3 Asymptotic refinement

Consider a statistic such as a Wald test statistic of a single restriction. Asymptotic methods are used to obtain an approximation to the cumulative distribution function of this statistic. For a statistic with a limiting normal distribution based on conventional first-order root- $N$  asymptotics, the approximation error behaves in the limit as a multiple of  $N^{-1/2}$  (so the error disappears as  $N \rightarrow \infty$ ). For example, a one-sided test with the nominal size of 0.05 will have the true size  $0.05 + O(N^{-1/2})$ , where  $O(N^{-1/2})$  behaves as a constant divided by  $\sqrt{N}$ .

Asymptotic methods with refinement have an approximation error that disappears at a faster rate. In particular, bootstraps with asymptotic refinement can implement second-order asymptotics that yield an approximation error that behaves as a multiple of  $N^{-1}$ . So a one-sided test with a nominal size of 0.05 now has a true size of  $0.05 + O(N^{-1})$ . This improvement is only asymptotic and is not guaranteed to exist in small samples. But simulation studies usually find that the improvement carries over to small samples.

We present bias-corrected (BC) confidence intervals that provide asymptotic refinement in sections [12.3.6–12.3.8](#). Hypothesis tests and confidence intervals based on the percentile-*t* method that provide asymptotic refinement are presented in section [12.5](#), and specialization to the wild cluster bootstrap is presented in section [12.6](#).

#### 12.2.4 Use the bootstrap with caution

Caution is needed in applying the bootstrap because it is easy to misapply. For example, one can always compute  $\text{se}_{\text{Boot}}(\hat{\theta})$  by using the formula in [\(12.1\)](#). But this estimate is inconsistent if, for example, the bootstrap resampling scheme assumes independent observations when observations are in fact correlated. And in some cases,  $\text{Var}(\hat{\theta})$  does not exist, even asymptotically. Then  $\text{se}_{\text{Boot}}(\hat{\theta})$  is estimating a nonexistent standard deviation.

The bootstraps presented in this chapter assume independence of observations or of clusters of observations. This does permit dependence via clustering, provided that observations are combined into clusters that are independent and the bootstrap is over the clusters. Then the bootstrap commands given in this chapter should include the `cluster(varlist)` option, where *varlist* denotes the clustering variables. The `idcluster(newvar)` option may additionally be needed; see section [12.3.5](#).

The bootstraps also assume that the estimator is a smooth estimator that is root-*N* consistent and asymptotically normal distributed. Some variations of the bootstrap can be applied to more complicated cases than this, but one should first read relevant journal articles. In particular, care is needed for estimators with a nonparametric component, for nonsmooth estimators, and for dependent data.

The Stata defaults for the number of bootstrap replications are set very low to speed up computation time. These values may be adequate for exploratory data analysis but should be greatly increased for published results; see section [12.3.4](#). And for published results, the seed should be set, using `set seed`, rather than using a sequence initially set by the default value used by Stata, to enable replication.

## 12.3 Bootstrap pairs using the vce(bootstrap) option

The most common use of the bootstrap is to obtain a consistent estimate of the standard errors of an estimator, with no asymptotic refinement. With standard Stata estimation commands, this can be easily done by using the `vce(bootstrap)` option.

### 12.3.1 Bootstrap-pairs method to estimate the variance–covariance matrix of the estimator

Let  $\mathbf{w}_i$  denote all the data for the  $i$ th observation. Most often,  $\mathbf{w}_i = (y_i, \mathbf{x}_i)$ , where  $y$  is a scalar dependent variable and  $\mathbf{x}$  is a regressor vector. More generally,  $\mathbf{w}_i = (\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i)$ , where now there may be several dependent variables and  $\mathbf{z}$  denotes instruments. We assume  $\mathbf{w}_i$  is i.i.d. over  $i$ .

Stata uses the following bootstrap-pairs algorithm:

1. Repeat steps a and b  $B$  independent times:
  - a. Draw a bootstrap sample of size  $N$  by sampling with replacement from the original data  $\mathbf{w}_1, \dots, \mathbf{w}_N$ . Denote the bootstrap sample by  $\mathbf{w}_1^*, \dots, \mathbf{w}_N^*$ .
  - b. Calculate an estimate,  $\hat{\boldsymbol{\theta}}^*$ , of  $\boldsymbol{\theta}$  based on  $\mathbf{w}_1^*, \dots, \mathbf{w}_N^*$ .
2. Given the  $B$  bootstrap estimates, denoted by  $\hat{\boldsymbol{\theta}}_1^*, \dots, \hat{\boldsymbol{\theta}}_B^*$ , the bootstrap estimate of the variance–covariance matrix of the estimator (VCE) is

$$\widehat{V}_{\text{Boot}}(\hat{\boldsymbol{\theta}}) = \frac{1}{B-1} \sum_{b=1}^B \left( \hat{\boldsymbol{\theta}}_b^* - \bar{\hat{\boldsymbol{\theta}}}^* \right) \left( \hat{\boldsymbol{\theta}}_b^* - \bar{\hat{\boldsymbol{\theta}}}^* \right)'$$

where  $\bar{\hat{\boldsymbol{\theta}}}^* = B^{-1} \sum_{b=1}^B \hat{\boldsymbol{\theta}}_b^*$ .

The corresponding standard-error estimate of the  $j$ th component of  $\hat{\boldsymbol{\theta}}$  is then

$$\text{se}_{\text{Boot}} \left( \hat{\theta}_j \right) = \left\{ \widehat{V}_{\text{Boot},jj}(\hat{\theta}) \right\}^{1/2}$$

The bootstrap resamples differ in the number of occurrences of each observation. For example, the first observation may appear twice in the first bootstrap sample, zero times in the second sample, once in the third sample, once in the fourth sample, and so on.

The method is called bootstrap pairs or paired bootstrap because in the simplest case  $\mathbf{w}_i = (y_i, \mathbf{x}_i)$  and the pair  $(y_i, \mathbf{x}_i)$  is being resampled. It is also called a case bootstrap because all the data for the  $i$ th case are resampled. And it is called a nonparametric bootstrap because no information about the conditional distribution of  $y_i$  given  $\mathbf{x}_i$  is used. For cross-sectional estimation commands, this bootstrap gives the same standard errors as those obtained by using the `vce(robust)` option if  $B \rightarrow \infty$ , aside from possible differences due to degrees-of-freedom correction that disappear for large  $N$ .

This bootstrap method is easily adapted to cluster pairs bootstraps. Then  $\mathbf{w}_i$  becomes  $\mathbf{w}_c$ , where  $c = 1, \dots, C$  denotes each of the  $C$  clusters, data are independent over  $c$ , resampling is over clusters, and the bootstrap resample is of size  $C$  clusters.

### 12.3.2 The `vce(bootstrap)` option

The bootstrap-pairs method to estimate the VCE can be obtained for most Stata cross-sectional estimation commands by using the estimator command option,

```
vce(bootstrap [ , bootstrap_options ])
```

which is often abbreviated to `vce(boot)`. We list many of the options in section [12.4.1](#) and illustrate some of the options in the following example.

The `vce(bootstrap)` option is also available for most panel-data `xt` estimation commands. The default bootstrap is then actually a cluster bootstrap over individuals  $i$ , rather than one over the individual observations  $(i, t)$ .

### 12.3.3 Bootstrap standard-errors example

We demonstrate the bootstrap using the same data on doctor visits (`docvis`) as those introduced in chapter 10, except that we use 1 regressor (`chronic`) and just the first 50 observations. This keeps output short, reduces computation time, and restricts attention to a small sample where the gains from asymptotic refinement may be greater.

```
. * Sample is 50 observations and a few variables from chapter 10 data
. qui use mus212bootdata
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	50	4.12	7.82106	0	43
age	50	4.162	1.160382	2.6	6.2
chronic	50	.28	.4535574	0	1

For standard error computation, we set the number of bootstrap replications to 400. We have

```
. * Option vce(bootstrap) to compute bootstrap standard errors
. poisson docvis chronic, vce(bootstrap, reps(400) seed(10101) nodots)
Poisson regression
Number of obs = 50
Replications = 400
Wald chi2(1) = 3.33
Prob > chi2 = 0.0679
Pseudo R2 = 0.0917
Log likelihood = -238.75384
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
docvis						
chronic	.9833014	.5386575	1.83	0.068	-.0724478	2.039051
_cons	1.031602	.3536507	2.92	0.004	.338459	1.724744

The output is qualitatively the same as that obtained by using any other method of standard error estimation. Quantitatively, however, the standard errors change, leading to different test statistics,  $z$ -values, and  $p$ -values. For `chronic`, the standard error of 0.539 is similar to the robust estimate of 0.515 given in the last column of the results from `estimates table` in the next section. Both standard errors control for Poisson overdispersion and are much larger than the default standard errors from `poisson`.

### 12.3.4 How many bootstraps?

More is better because when  $B = \infty$ , the same results are obtained regardless of the initial seed. The Stata default is to perform 50 bootstrap replications to minimize computation time. This value may be useful during the modeling cycle, but for final results given in an article, this value is much too low.

At a time of much less computing power, [Efron and Tibshirani \(1993, 52\)](#) stated that for standard error estimation “ $B = 50$  is often enough to give a good estimate” and “very seldom are more than  $B = 200$  replications needed.” [Andrews and Buchinsky \(2000\)](#) show that the bootstrap estimate of the standard error of  $\hat{\theta}$  with  $B = 384$  is within 10% of that with  $B = \infty$  with a probability of 0.95, in the special case that  $\hat{\theta}$  has no excess kurtosis. The community-contributed `bssize` command ([Poi 2004](#)) performs the calculations needed to implement the methods of [Andrews and Buchinsky \(2000\)](#). In this book, we often use  $B = 400$  for bootstrap estimation of standard errors. For published research,  $B$  should be much higher than this.

For bootstrap confidence intervals and hypothesis tests,  $B$  needs to be even higher than for standard error regressions because confidence levels and test sizes use tails of the distribution of the estimator. For tests at the  $\alpha$  level or at  $100(1 - \alpha)\%$  confidence intervals, there are reasons for choosing  $B$  so that  $\alpha(B + 1)$  is an integer. In subsequent analysis, we use  $B = 999$  for confidence intervals and hypothesis tests when  $\alpha = 0.05$ . Ideally, published results use a much higher value, such as  $B = 9999$ , so that the value of the seed has little effect on results.

To see the effects of the number of bootstraps on standard error estimation, we compare results with very few bootstraps,  $B = 50$ , using two different seeds, and with many bootstraps,  $B = 2000$ , and two different seeds. We also present the robust standard error obtained by using the `vce(robust)` option. We have

```

. * Bootstrap standard errors for different reps and seeds
. qui poisson docvis chronic, vce(bootstrap, reps(50) seed(10101))
. estimates store boot50
. qui poisson docvis chronic, vce(bootstrap, reps(50) seed(20202))
. estimates store boot50diff
. qui poisson docvis chronic, vce(bootstrap, reps(2000) seed(10101))
. estimates store b2000
. qui poisson docvis chronic, vce(bootstrap, reps(2000) seed(20202))
. estimates store b2000diff
. qui poisson docvis chronic, vce(robust)
. estimates store robust
. estimates table boot50 boot50diff b2000 b2000diff robust, b(%8.5f) se(%8.5f)

```

Variable	boot50	boot50^f	b2000	b2000d^f	robust
chronic	0.98330	0.98330	0.98330	0.98330	0.98330
	0.45444	0.59923	0.54178	0.53413	0.51549
_cons	1.03160	1.03160	1.03160	1.03160	1.03160
	0.37131	0.37533	0.36414	0.36428	0.34467

Legend: b/se

Comparing the two replications with  $B = 50$  but different seeds, we see the standard error of `chronic` differs by 15% (0.454 versus 0.599). For  $B = 2000$ , the bootstrap standard errors still differ from each other (0.542 versus 0.534) and also from the robust standard errors (0.515) partly because of the use of the multiplicative finite-sample adjustment  $N/(N - K)$  with  $N = 50$  in calculating robust standard errors.

### 12.3.5 Cluster pairs bootstrap

For cross-sectional estimation commands, the `vce(bootstrap)` option performs a paired bootstrap that assumes independence over  $i$ . The bootstrap resamples are obtained by sampling from the individual observations with replacement.

The data may instead be clustered, with observations correlated within cluster and independent across clusters. The `vce(bootstrap, cluster(varlist))` option performs a cluster bootstrap that samples the

clusters with replacement. If there are  $C$  clusters, then the bootstrap resample has  $C$  clusters. This may mean that the number of observations  $N = \sum_{c=1}^C N_c$  may vary across bootstrap resamples, but this poses no problem.

As an example,

Poisson regression						
Log likelihood = -238.75384						
(Replications based on 26 clusters in age)						
docvis	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
chronic	.9833014	.4805585	2.05	0.041	.0414241	1.925179
_cons	1.031602	.2936567	3.51	0.000	.4560451	1.607158

The cluster-pairs bootstrap estimate of the standard error of  $\beta_{\text{chronic}}$  is 0.481. If we instead obtain the usual (nonbootstrap) cluster-robust standard errors, using the `vce(cluster age)` option, the cluster estimate of the standard error is 0.449.

In this somewhat manufactured example, the cluster-robust standard errors are similar to the heteroskedastic-robust standard errors. In practice, cluster-robust standard errors can be much larger if errors are clustered; see section [3.4.6](#).

The asymptotic theory for inference with clustered data is based on the assumption that the number of clusters  $G \rightarrow \infty$ . In many applications, there may be few clusters; for example, clustering may be on region with only 10 regions. Then there can be considerable size distortion in hypothesis tests using either cluster-robust standard errors or cluster-bootstrap standard errors. Then one should use the wild cluster bootstrap presented in section [12.6](#), which provides an asymptotic refinement.

Some applications use cluster identifiers in computing estimators. For example, suppose cluster-specific indicator variables (fixed effects) are

directly included as regressors. This can be done, for example, by using factor-variable notation and the regressors `i.id`, where *id* is the cluster identifier. If the first cluster in the original sample appears twice in a cluster-bootstrap resample, then its cluster dummy will be nonzero twice in the resample, rather than once, and the cluster dummies will no longer be unique to each observation in the resample. For the bootstrap resample, we should instead define a new set of  $C$  unique cluster dummies that will each be nonzero exactly once. The `idcluster(newvar)` option does this, creating a new variable containing a unique identifier for each observation in the resampled cluster. This is particularly relevant for estimation with fixed effects, including fixed-effects panel-data estimators.

For most `xt` commands, the `vce(bootstrap)` option default is to perform a cluster bootstrap with clustering on the cluster identifier specified in the `xtset` command. For `xt` commands with the `fe` option, there is no need to use the `idcluster()` option because the `xt` command is coded to account for the preceding complication.

For estimation commands without the `vce(bootstrap, cluster())` option, one can instead use the `bootstrap` prefix with option `cluster()`; see section [12.4.1](#).

### 12.3.6 Bootstrap confidence intervals

The output after a command with the `vce(bootstrap)` option includes a “normal-based” 95% confidence interval for  $\theta$  that equals

$$\left[ \widehat{\theta} - 1.96 \times \text{se}_{\text{Boot}}(\widehat{\theta}), \widehat{\theta} + 1.96 \times \text{se}_{\text{Boot}}(\widehat{\theta}) \right]$$

and is a standard Wald asymptotic confidence interval, except that the bootstrap is used to compute the standard error.

Additional confidence intervals can be obtained by using the `estat bootstrap` postestimation command, defined in the next section.

The percentile method uses the relevant percentiles of the empirical distribution of the  $B$  bootstrap estimates  $\widehat{\theta}_1^*, \dots, \widehat{\theta}_B^*$ . In particular, a percentile 95% confidence interval for  $\theta$  is

$$\left[ \widehat{\theta}_{0.025}^*, \widehat{\theta}_{0.975}^* \right]$$

ranging from the 2.5th percentile to the 97.5th percentile of  $\widehat{\theta}_1^*, \dots, \widehat{\theta}_B^*$ . This confidence interval has the advantage of being asymmetric around  $\widehat{\theta}$  and being invariant to monotonic transformation of  $\theta$ . Like the normal-based confidence interval, it does not provide an asymptotic refinement, but there are still theoretical reasons to believe it provides a better approximation than the normal-based confidence interval.

The validity of bootstrap confidence intervals and tests requires convergence of the bootstrap distribution. The validity of bootstrap standard errors requires stronger uniform integrability conditions because convergence in distribution does not imply convergence in moments. So the bootstrap percentile method requires weaker assumptions than the “normal-based” method, which uses the bootstrap standard errors.

The BC method is a modification of the percentile method that incorporates a bootstrap estimate of the finite-sample bias in  $\widehat{\theta}$ . For example, if the estimator is upward biased, as measured by estimated median bias, then the confidence interval is moved to the left. So if 40%, rather than 50%, of  $\widehat{\theta}_1^*, \dots, \widehat{\theta}_B^*$  are less than  $\widehat{\theta}$ , then a BC 95% confidence interval might use  $[\widehat{\theta}_{0.007}^*, \widehat{\theta}_{0.927}^*]$ , say, rather than  $[\widehat{\theta}_{0.025}^*, \widehat{\theta}_{0.975}^*]$ .

The bias-corrected accelerated (BCa) confidence interval is an adjustment to the BC method that adds an “acceleration” component that permits the asymptotic variance of  $\widehat{\theta}$  to vary with  $\theta$ . This requires the use of a jackknife that can add considerable computational time and is not possible for all estimators. The formulas for BC and BCa confidence intervals are given in [R] **bootstrap** and in books such as [Efron and Tibshirani \(1993, 185\)](#) and [Davison and Hinkley \(1997, 204\)](#).

The BCA confidence interval has the theoretical advantage over the other confidence intervals that it offers an asymptotic refinement, defined in section [12.2.3](#). So a BCA 95% confidence interval has a coverage rate of  $0.95 + O(N^{-1})$ , compared with  $0.95 + O(N^{-1/2})$  for the other methods.

An alternative, quite general method to obtain an asymptotic refinement is to use the percentile-*t* method, presented in section [12.5.1](#) and applied in section [12.6](#).

In practice, microeconomics studies use the percentile-*t* method if asymptotic refinement is desired; the BC and BCA confidence intervals are seldom used. The `estat bootstrap` command does not provide percentile-*t* confidence intervals, but these can be obtained by using the `bootstrap` prefix, as we demonstrate in section [12.5.3](#), and for a wild bootstrap by using the user-provided `boottest` command that is presented in section [12.6.2](#).

### 12.3.7 The `estat bootstrap` postestimation command

The `estat bootstrap` command can be issued after an estimation command that has the `vce(bootstrap)` option or after the `bootstrap` prefix. The syntax for `estat bootstrap` is

```
estat bootstrap [ , options ]
```

where the options include `normal` for normal-based confidence intervals, `percentile` for percentile-based confidence intervals, `bc` for BC confidence intervals, and option `bca` for BCA confidence intervals. If you want to use the `bca` option, the preceding `bootstrap` must be done with the `bca` option specified to perform the necessary additional jackknife computation. The `all` option provides all available confidence intervals.

### 12.3.8 Bootstrap confidence-intervals example

We obtain these various confidence intervals for the Poisson example. To obtain the BCA interval, we must use the `bca` option with the original bootstrap. To speed up bootstraps, we should include only necessary variables in the dataset. For reasonable bootstrap precision, we set  $B = 999$ .

We have

```
. * Bootstrap confidence intervals: Normal based, percentile, BC, and BCa
. qui poisson docvis chronic, vce(bootstrap, reps(999) seed(10101) bca)
. estat bootstrap, all

Poisson regression
Number of obs      =          50
Replications       =         999
```

docvis	Observed coefficient	Bias	Bootstrap std. err.	[95% conf. interval]		
chronic	.98330144	.0132307	.54137854	-.077781	2.044384	(N)
				-.0139438	2.061742	(P)
				-.079079	2.019438	(BC)
				.0295944	2.08349	(BCa)
_cons	1.0316016	-.0769223	.35685342	.3321817	1.731021	(N)
				.186586	1.582409	(P)
				.268264	1.641356	(BC)
				.386773	1.771351	(BCa)

Key: N: Normal  
P: Percentile  
BC: Bias-corrected  
BCa: Bias-corrected and accelerated

The confidence intervals for  $\beta_{\text{chronic}}$  are, respectively,  $[-0.08, 2.04]$ ,  $[-0.014, 2.06]$ ,  $[-0.08, 2.02]$ , and  $[0.03, 2.08]$ . The differences here are not great. Only the normal-based confidence interval is symmetric about  $\widehat{\beta}_{\text{chronic}}$ .

### 12.3.9 Bootstrap estimate of bias

Suppose that the estimator  $\widehat{\theta}$  is biased for  $\theta$ . Let  $\overline{\widehat{\theta}}^*$  be the average of the  $B$  bootstraps and  $\widehat{\theta}$  be the estimate from the original model. Note that  $\overline{\widehat{\theta}}^*$  is not an unbiased estimate of  $\theta$ . Instead, the difference  $\overline{\widehat{\theta}}^* - \widehat{\theta}$  provides a bootstrap estimate of the bias of the estimate  $\widehat{\theta}$ . The bootstrap views the data-generating process (DGP) value as  $\widehat{\theta}$ , and  $\overline{\widehat{\theta}}^*$  is viewed as the mean of the estimator given this DGP value.

Below, we list `e(b_bs)`, which contains the average of the bootstrap estimates.

```
. matrix list e(b_bs)
e(b_bs)[1,2]
    docvis:      docvis:
    chronic      _cons
y1   .99653216   .95467931
```

The above output indicates that  $\bar{\theta}^* = 0.9965$ ; the output from `estat bootstrap, all` indicates that  $\hat{\theta} = 0.9833$ . Thus, the bootstrap estimate of bias is 0.0132, which is reported in the `estat bootstrap, all` output. Because  $\hat{\theta} = 0.9833$  is upward biased by 0.0132, we must subtract this bias to get a BC estimate of  $\theta$  that equals  $0.9833 - 0.0132 = 0.9701$ . Such BC estimates are not used often, however, because the bootstrap estimate of mean bias is a very noisy estimate; see [Efron and Tibshirani \(1993, 138\)](#).

## 12.4 Bootstrap pairs using the `bootstrap` command

The `bootstrap` prefix can be applied to a wide range of Stata commands such as nonestimation commands, community-contributed commands, two-step estimators, and Stata estimators without the `vce(bootstrap)` option. Before doing so, the user should verify that the estimator is one for which it is appropriate to apply the bootstrap; see section [12.2.4](#).

### 12.4.1 The `bootstrap` prefix

The syntax for the `bootstrap` prefix is

```
bootstrap explist [ , options eform_option ] : command
```

The command being bootstrapped can be an estimation command, other commands such as `summarize`, or community-contributed commands. The argument *explist* provides the quantity or quantities to be bootstrapped. These can be one or more expressions, possibly given names [so `newvar = (exp)`].

For estimation commands, not setting *explist* or setting *explist* to `_b` leads to a bootstrap of the parameter estimates. Setting *explist* instead to `_se` leads to a bootstrap of the standard errors of the parameter estimates. Thus `bootstrap: poisson y x` bootstraps parameter estimates, as does `bootstrap _b: poisson y x`. The `bootstrap _se: poisson y x` command instead bootstraps the standard errors. The `bootstrap _b[x]: poisson y x` command bootstraps just the coefficient of `x` and not that of the intercept. The `bootstrap bx=_b[x]: poisson y x` command does the same, with the results named `bx` rather than with the default name of `_bs_1`.

The options include `reps(#)` to set the number of bootstrap replications; `seed(#)` to set the random-number generator seed value to enable reproducibility; `nodots` to suppress dots produced for each bootstrap replication; `group(varname)`, which may be needed along with `idcluster()`; `strata(varlist)` for bootstrap over strata; `size(#)` to draw samples of size `#`; `bca` to compute the acceleration for a BCA confidence

interval; and `saving()` to save results from each bootstrap iteration in a file. The `eform_option` option enables bootstraps for  $e^\theta$  rather than  $\theta$ .

A cluster pairs bootstrap can be performed using the `cluster(varlist)` option; the `idcluster(newvar)` option is needed for some cluster bootstraps (see section [12.3.5](#)). For panel data with identifiers for both individual and time set by the `xtset` command, the cluster bootstrap can then fail with error message

```
repeated time values within panel  
the most likely cause for this error is misspecifying the cluster(), idcluster(),  
or group() option
```

To avoid this, one can `xtset` just the individual identifier.

If `bootstrap` is applied to commands other than Stata estimation commands, it produces a warning message. For example, the user-written `poissrobust` command defined below, leads to the warning

```
Warning: Since poissrobust is not an estimation command or does not set  
e(sample), bootstrap has no way to determine which observations are  
used in calculating the statistics and so assumes that all  
observations are used. This means no observations will be excluded  
from the resampling because of missing values or other reasons.
```

If the assumption is not true, press Break, save the data, and drop the observations that are to be excluded. Be sure that the dataset in memory contains only the relevant data.

Because we know that this is not a problem in the examples below and we want to minimize output, we use the `nowarn` option to suppress this warning.

The output from `bootstrap` includes the bootstrap estimate of the standard error of the statistic of interest and the associated normal-based 95% confidence interval. The `estat bootstrap` command after `bootstrap` can also compute percentile, BC and BCA confidence intervals. For brevity, we do not obtain these additional confidence intervals in the examples below.

#### 12.4.2 Bootstrap parameter estimate from a Stata estimation command

The `bootstrap` prefix is easily applied to an existing Stata estimation command. It gives exactly the same result as given by directly using the

Stata estimation command with the `vce(bootstrap)` option, if this option is available and the same values of  $B$  and the seed are used.

We illustrate this for doctor visits. Because we are bootstrapping parameter estimates from an estimation command, there is no need to provide *explist*.

```
. * bootstrap command applied to Stata estimation command
. bootstrap, reps(400) seed(10101) nodots noheader: poisson docvis chronic
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
docvis						
chronic	.9833014	.5386575	1.83	0.068	-.0724478	2.039051
_cons	1.031602	.3536507	2.92	0.004	.338459	1.724744

The results are exactly the same as those obtained in section [12.3.3](#) by using `poisson` with the `vce(bootstrap)` option.

### 12.4.3 Bootstrap standard error from a Stata estimation command

$\hat{\theta}$  is not an exact estimate of  $\theta$ , and, in addition,  $se(\hat{\theta})$  is not an exact estimate of the standard deviation of the estimator  $\hat{\theta}$ . We consider a bootstrap of the standard error,  $se(\hat{\theta})$ , to obtain an estimate of the standard error of  $se(\hat{\theta})$ .

We bootstrap both the coefficients and their standard errors. We have

```
. * Bootstrap estimate of the standard error of a coefficient estimate
. bootstrap _b _se, reps(400) seed(10101) nodots: poisson docvis chronic
```

Bootstrap results	Number of obs = 50 Replications = 400
-------------------	--

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
docvis						
chronic	.9833014	.5386575	1.83	0.068	-.0724478	2.039051
_cons	1.031602	.3536507	2.92	0.004	.338459	1.724744
docvis_se						
chronic	.1393729	.0236361	5.90	0.000	.0930471	.1856987
_cons	.0995037	.0202009	4.93	0.000	.0599106	.1390968

The bootstrap reveals that there is considerable noise in  $\text{se}(\hat{\beta}_{\text{chronic}})$ , with an estimated standard error of 0.024 and the 95% confidence interval [0.093, 0.186].

Ideally, the bootstrap standard error of  $\hat{\beta}$ , here 0.539, should be close to 0.139, the estimated  $\text{se}(\hat{\beta})$  from the original estimation sample. And it should be close to the average of the  $\text{se}(\hat{\beta})$ 's from the 400 bootstraps that from additional code not given is 0.150. The large difference is a clear sign of problems in the method used to obtain  $\text{se}(\hat{\beta}_{\text{chronic}})$ . The problem is that the default Poisson standard errors were used in `poisson` above, and given the large overdispersion, these standard errors are very poor. If we repeated the exercise with `poisson` and the `vce(robust)` option, this difference disappears because the robust standard error is 0.515.

#### 12.4.4 Bootstrap standard error from a user-written estimation command

Continuing the previous example, we would like an estimate of the properties of the robust standard errors after Poisson regression. This can be obtained by using `poisson` with the `vce(robust)` option in the preceding bootstrap. We instead use an alternative approach that can be applied in a wide range of settings.

We write a program named `poissrobust` that returns the Poisson maximum-likelihood estimator (MLE) estimates in `b` and the robust estimate of the VCE of the Poisson MLE in `v`. Then, we apply the `bootstrap` prefix to `poissrobust` rather than to `poisson, vce(robust)`.

Because we want to return `e` and `v`, the program must be `eclass`. The program is

```

* Program to return b and robust estimate V of the VCE
program poissrobust, eclass
    version 17
    tempname b V
    poisson docvis chronic, vce(robust)
    matrix `b' = e(b)
    matrix `V' = e(V)
    ereturn post `b' `V'
end

```

Next, it is good practice to check the program, typing the commands

```

. * Check preceding program by running once
. poissrobust
    (output omitted)
. ereturn display
    (output omitted)

```

The omitted output is the same as that from `poisson, vce(robust)`.

We then bootstrap 400 times. The bootstrap estimate of the standard error of  $\text{se}(\hat{\theta})$  is the standard deviation of the  $B$  values of  $\text{se}(\hat{\theta})$ . We have

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
docvis					Number of obs = 50	
	chronic	.9833014	.5386575	1.83	0.068	-.0724478 2.039051
docvis_se	_cons	1.031602	.3536507	2.92	0.004	.338459 1.724744
	chronic	.5154894	.0772422	6.67	0.000	.3640974 .6668814
	_cons	.3446734	.065348	5.27	0.000	.2165936 .4727532

There is considerable noise in the robust standard error, with the standard error of  $\text{se}(\hat{\beta}_{\text{chronic}})$  equal to 0.077 and a 95% confidence interval of [0.364, 0.667]. The upper limit is about twice the lower limit, as was the case for the default standard error. In other examples, robust standard errors can be much less precise than default standard errors.

## 12.4.5 Bootstrap two-step estimator

The preceding method of applying the `bootstrap` prefix to a user-defined estimation command can also be applied to a two-step estimator.

A sequential two-step estimator of, say,  $\hat{\beta}$  is one that depends in part on a consistent first-stage estimator, say,  $\hat{\alpha}$ . In some examples—notably, feasible generalized least squares, where  $\hat{\alpha}$  denotes error variance parameters—one can do regular inference, ignoring any estimation error in  $\hat{\alpha}$ . More generally, however, the asymptotic distribution of  $\hat{\beta}$  will depend on that of  $\hat{\alpha}$ . Asymptotic results do exist that confirm the asymptotic normality of leading examples of two-step estimators and provide a general formula for  $\text{Var}(\hat{\beta})$ . But this formula is usually complicated, both analytically and in implementation. A much simpler method is to use the bootstrap, which is valid if indeed the two-step estimator is known to be asymptotically normal.

A leading example is Heckman's two-step estimator in the selection model; see section 19.6.4. We use the same example as in that section. We first read in the data and form the dependent variable `dy` and the regressor list given in `xlist`.

```
. * Set up the selection model two-step estimator data of the tobit chapter
. qui use mus219mepsambexp, clear
. global xlist age female educ blhisp totchr ins
```

The following program produces the Heckman two-step estimator:

```
* Program to return b for Heckman two-step estimator of selection model
program hecktwostep, eclass
    version 17
    tempname b V
    tempvar xb
    capture drop invmills
    probit dy $xlist
    predict `xb', xb
    generate invmills = normalden(`xb')/normal(`xb')
    regress lny $xlist invmills if dy==1
    matrix `b' = e(b)
    ereturn post `b'
end
```

This program can be checked by typing `hecktwostep` in isolation. This leads to the same parameter estimates as in section 19.6.4. Here  $\beta$  denotes the second-stage regression coefficients of regressors and the inverse of the Mills ratio. The inverse of the Mills ratio depends on the first-stage probit parameter estimates  $\hat{\alpha}$ .

To obtain correct standard errors that control for the two-step estimation, we bootstrap, where the bootstrap is resampling all data used in both steps.

```
. * Bootstrap for Heckman two-step estimator using tobit chapter example
. bootstrap _b, reps(400) seed(10101) nodots nowarn: hecktwostep
Bootstrap results                                         Number of obs = 3,328
                                                               Replications = 400
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
age	.202124	.024179	8.36	0.000	.154734	.2495141
female	.2891575	.0667082	4.33	0.000	.1584118	.4199032
educ	.0119928	.0112873	1.06	0.288	-.0101298	.0341154
blhisp	-.1810582	.0622552	-2.91	0.004	-.3030762	-.0590402
totchr	.4983315	.0418504	11.91	0.000	.4163063	.5803568
ins	-.0474019	.0511954	-0.93	0.354	-.1477431	.0529393
invmills	-.4801696	.2696967	-1.78	0.075	-1.008766	.0484263
_cons	5.302572	.2753108	19.26	0.000	4.762973	5.842171

The standard errors are generally within 10% of those given in chapter 19, which are based on analytical results.

The bootstrap for a sequential two-step estimator has the advantage of convenience but can require considerable computational time. An alternative way to obtain the standard errors is to stack the equations for each step and estimate jointly by (just-identified) generalized methods of moments; see section 19.6.4 for this approach.

## 12.4.6 Bootstrap Hausman test

The Hausman test statistic, presented in section [11.9.5](#), is

$$H = \left( \widehat{\theta} - \widetilde{\theta} \right)' \left\{ \widehat{V} \left( \widehat{\theta} - \widetilde{\theta} \right) \right\}^{-1} \left( \widehat{\theta} - \widetilde{\theta} \right) \stackrel{a}{\sim} \chi^2(h) \text{ under } H_0$$

where  $\hat{\theta}$  and  $\tilde{\theta}$  are different estimators of  $\theta$ .

Standard implementations of the Hausman test, including the `hausman` command presented in section [11.9.5](#), require that one of the estimators be fully efficient under  $H_0$ . Great simplification occurs because

$\text{Var}(\hat{\theta} - \tilde{\theta}) = \text{Var}(\tilde{\theta}) - \text{Var}(\hat{\theta})$  if  $\hat{\theta}$  is fully efficient under  $H_0$ . For some likelihood-based estimators, correct model specification is necessary for consistency, and in that case the estimator is also fully efficient. But often it is standard to not require that the estimator be efficient. In particular, if there is reason to use robust standard errors, then the estimator is not fully efficient.

The bootstrap can be used to estimate  $\text{Var}(\hat{\theta} - \tilde{\theta})$ , without the need to assume that one of the estimators is fully efficient under  $H_0$ . The  $B$  replications yield  $B$  estimates of  $\hat{\theta}$  and  $\tilde{\theta}$  and hence of  $\hat{\theta} - \tilde{\theta}$ . We estimate  $\text{Var}(\hat{\theta} - \tilde{\theta})$  with  $\{1/(B-1)\} \sum_b (\hat{\theta}_b - \tilde{\theta}_b - \bar{\theta}_{\text{diff}}^*) (\hat{\theta}_b - \tilde{\theta}_b - \bar{\theta}_{\text{diff}}^*)'$ , where  $\bar{\theta}_{\text{diff}}^* = (1/B) \sum_b (\hat{\theta}_b - \tilde{\theta}_b)$ .

As an example, we consider a Hausman test for endogeneity of a regressor based on comparing instrumental-variables (IV) and ordinary least-squares (OLS) estimates. Large values of  $H$  lead to rejection of the null hypothesis that all regressors are exogenous.

The following program is written for the two-stage least-squares example presented in section [7.4.6](#).

```
* Program to return (b1-b2) for Hausman test of endogeneity
program hausmantest, eclass
    version 17
    tempname b bols biv
    regress ldrugexp hi_empunion totchr age female blhispl linc, vce(robust)
    matrix `bols' = e(b)
    ivregress 2sls ldrugexp (hi_empunion = ssiratio) totchr age female blhispl ///
               linc
    matrix `biv' = e(b)
    matrix `b' = `bols' - `biv'
    ereturn post `b'
end
```

This program can be checked by typing `hausmantest` in isolation.

We then run the bootstrap.

```
. * Bootstrap estimates for Hausman test using IV chapter example
. qui use mus207mepspresdrugs, clear
. bootstrap _b, reps(400) seed(10101) nodots nowarn: hausmantest
Bootstrap results
Number of obs = 10,367
Replications = 400
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
hi_empunion	.8847442	.1876556	4.71	0.000	.5169459	1.252542
totchr	-.0090062	.0037012	-2.43	0.015	-.0162604	-.0017519
age	.0088707	.002019	4.39	0.000	.0049135	.0128278
female	.0710012	.016444	4.32	0.000	.0387715	.1032308
blhisp	.0602153	.0169786	3.55	0.000	.0269379	.0934928
linc	-.0699006	.0149192	-4.69	0.000	-.0991416	-.0406595
_cons	-.8461408	.1909968	-4.43	0.000	-1.220488	-.4717939

For the single potentially endogenous regressor, we can use the *t* statistic given above, or we can use the *test* command. The latter yields

```
. * Perform Hausman test on coefficient of the potentially endogenous regressor
. test hi_empunion
( 1) hi_empunion = 0
chi2( 1) = 22.23
Prob > chi2 = 0.0000
```

The null hypothesis of regressor exogeneity is strongly rejected.

The *test* command can also be used to perform a Hausman test based on all regressors. We have

```
. * Perform Hausman test on the coefficients of all regressors
. test hi_empunion totchr age female blhisp linc _cons
( 1) hi_empunion = 0
( 2) totchr = 0
( 3) age = 0
( 4) female = 0
( 5) blhisp = 0
( 6) linc = 0
( 7) _cons = 0
chi2( 7) = 23.08
Prob > chi2 = 0.0017
```

The preceding example has wide applicability for robust Hausman tests.

### 12.4.7 Bootstrap standard error of the coefficient of variation

The bootstrap need not be restricted to regression models. A simple example is to obtain a bootstrap estimate of the standard error of the sample mean of `docvis`. This can be obtained by using the `bootstrap _se: mean docvis` command.

A slightly more difficult example is to obtain the bootstrap estimate of the standard error of the coefficient of variation ( $= s_x / \bar{x}$ ) of doctor visits. The results stored in `r()` after `summarize` allow the coefficient of variation to be computed as `r(sd) / r(mean)`, so we bootstrap this quantity.

To do this, we use `bootstrap` with the expression `coeffvar=(r(sd) / r(mean))`. This bootstraps the quantity `r(sd) / r(mean)` and gives it the name `coeffvar`. We have

```
. * Bootstrap estimate of the standard error of the coefficient of variation
. qui use mus212bootdata, clear
. bootstrap coeffvar=(r(sd)/r(mean)), reps(400) seed(10101) nodots
>      nowarn: summarize docvis
Bootstrap results                                         Number of obs =  50
                                                               Replications  = 400
                                                              
Command: summarize docvis
coeffvar: r(sd)/r(mean)
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
coeffvar	1.898316	.269266	7.05	0.000	1.370564	2.426067

The normal-based bootstrap 95% confidence interval for the coefficient of variation is [1.37, 2.43].

## 12.5 Percentile-t bootstraps with asymptotic refinement

Asymptotic refinement, defined in section [12.2.3](#), can be obtained using BCA confidence intervals that are provided by the `estat bootstrap` postestimation command; see sections [12.3.6–12.3.8](#).

In this section, we present an alternative and more widely used method to obtain asymptotic refinement. The percentile- $t$  method has general applicability to hypothesis testing and confidence intervals. And it is the basis for the wild cluster bootstrap, presented in section [12.6](#), for improved inference when there are few clusters.

### 12.5.1 Percentile-t method

Standard asymptotic inference is based on the central limit theorem, which is based on a series expansion of the characteristic function that neglects higher-order terms. An asymptotic refinement can be obtained by using higher-order terms in an Edgeworth series expansion, a power series in  $N^{-1/2}$ . The percentile- $t$  bootstrap that bootstraps the  $t$  statistic is an empirical method that implements the Edgeworth expansion. A brief summary of the theory is given in [Cameron and Trivedi \(2005](#), chap. 11.4), and more detail is given in, for example, Efron and Tibshirani ([1993](#), chap. 22) and [Horowitz \(2001\)](#).

This approach to obtaining asymptotic refinement requires bootstrapping a quantity that is asymptotically pivotal, meaning that its asymptotic distribution does not depend on unknown parameters. The estimate  $\hat{\theta}$  is not asymptotically pivotal, because its variance depends on unknown parameters. Percentile methods therefore do not provide an asymptotic refinement unless an adjustment is made, notably, that by the BCA percentile method.

The  $t$  statistic is asymptotically pivotal, however, because its asymptotic distribution is the standard normal distribution, which has no unknown parameters. Percentile- $t$  methods or bootstrap- $t$  methods bootstrap the  $t$  statistic,

$$t = (\hat{\theta} - \theta_0) / \text{se}(\hat{\theta}) \quad (12.2)$$

where  $\theta_0$  is the null hypothesis value of  $\theta$ .

The bootstrap views the original sample as the DGP, so the bootstrap sets the DGP value of  $\theta$  to be  $\hat{\theta}$ . So in each bootstrap resample, we compute a  $t$  statistic centered on  $\hat{\theta}$ ,

$$t_b^* = (\hat{\theta}_b^* - \hat{\theta}) / \text{se}(\hat{\theta}_b^*) \quad (12.3)$$

where  $\hat{\theta}_b^*$  is the parameter estimate in the  $b$ th bootstrap and  $\text{se}(\hat{\theta}_b^*)$  is a consistent estimate of the standard error of  $\hat{\theta}_b^*$ , often a robust or cluster-robust standard error. As already noted,  $t_b^*$  is centered on  $\hat{\theta}$  because  $\hat{\theta}$  is the true DGP value in the bootstrap sample.

The  $B$  bootstraps yield the  $t$ -values  $t_1^*, \dots, t_B^*$ , whose empirical distribution is used as the estimate of the distribution of the  $t$  statistic.

For a one-sided test of  $\theta_0$ , we use either  $B^{-1} \sum_{b=1}^B \mathbf{1}(t_b^* < t)$  or  $B^{-1} \sum_{b=1}^B \mathbf{1}(t_b^* > t)$ .

For a two-sided test of  $H_0: \theta = 0$ , there are two ways to compute the  $p$ -value of the original test statistic  $t = \hat{\theta}/\text{se}(\hat{\theta})$ . A symmetric  $t$  test uses

$$p = \frac{1}{B} \sum_{b=1}^B \mathbf{1}(|t_b^*| > |t|) \quad (12.4)$$

which is the fraction of times in  $B$  replications that  $|t^*| > |t|$ . This method is the one necessarily used if we generalize to a test statistic that is always positive, such as a test statistic that is asymptotically chi-squared distributed.

An equal-tail two-sided test instead uses

$$p = 2 \min \left\{ \frac{1}{B} \sum_{b=1}^B \mathbf{1}(t_b^* < t), \frac{1}{B} \sum_{b=1}^B \mathbf{1}(t_b^* > t) \right\} \quad (12.5)$$

This rejects  $H_0$  at level  $\alpha$  if either a lower-tail test rejects at level  $\alpha/2$  or an upper-tail test rejects at level  $\alpha/2$ . This procedure is especially warranted when rejection in one tail is more likely than in the other tail, such as for a Wald test based on the two-stage least squares (2SLS) estimator, which is biased when instruments are weak instruments.

There are several ways to form confidence intervals. A percentile- $t$  symmetric 95% confidence interval for  $\theta$  uses

$$\left[ \hat{\theta} - |t^*|_{0.95} \times \text{se}(\hat{\theta}), \hat{\theta} + |t^*|_{0.95} \times \text{se}(\hat{\theta}) \right]$$

where  $|t^*|_{0.95}$  is the 95th percentile of the distribution of  $|t_1^*|, \dots, |t_B^*|$ .

One variant of an equal-tailed percentile- $t$  95% confidence interval for  $\theta$  is

$$\left[ \hat{\theta} - t_{0.975}^* \times \text{se}(\hat{\theta}), \hat{\theta} - t_{0.025}^* \times \text{se}(\hat{\theta}) \right] \quad (12.6)$$

See, for example, [Efron and Tibshirani \(1993, 173–174\)](#) or [Hansen \(2022a, 283–284\)](#).

With the definition in (12.6), the confidence interval can include 0, even though we would reject  $H_0 : \theta = 0$  using the equal-tail two-sided test in (12.5). An alternative equal-tailed percentile- $t$  95% confidence interval is therefore obtained by inverting the equal-tailed two-sided test defined in

(12.5); see section 11.3.13 for obtaining a confidence interval by inverting a test statistic.

### 12.5.2 Percentile-t Wald test

Stata does not automatically produce the percentile-*t* method. In this section, we present a general method to implement the percentile-*t* method using the `bootstrap` prefix with bootstrap pairs resampling. Section 12.6 then presents the community-contributed `boottest` command, which implements the percentile-*t* method using wild bootstrap resampling.

We continue with a count regression of `docvis` on `chronic`. A complication is that the standard error given in either (12.2) or (12.3) needs to be a consistent estimate of the standard deviation of the estimator. So we use `bootstrap` to perform a bootstrap of `poisson`, where the VCE is estimated with the `vce(robust)` option, rather than using the default Poisson standard-error estimates that are greatly downward biased.

We store the sample parameter estimate and standard error as local macros before bootstrapping the *t* statistic given in (12.3). The 999 bootstrap values  $t_1^*, \dots, t_{999}^*$  are saved as variable `tstar` in `percentilet.dta`.

```
. * Percentile-t for a single coefficient: Bootstrap the t statistic
. qui use mus212bootdata, clear
. qui poisson docvis chronic, vce(robust)
. local theta = _b[chronic]
. local setheta = _se[chronic]

. bootstrap tstar=((_b[chronic]-`theta')/_se[chronic]), seed(10101) nodots
>     reps(999) saving(percentilet, replace): poisson docvis chronic,
>     vce(robust)
(file percentilet.dta not found)

Bootstrap results                                         Number of obs = 50
                                                               Replications = 999

Command: poisson docvis chronic, vce(robust)
          tstar: (_b[chronic]-.9833014421442415)/_se[chronic]
```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
tstar	0	1.288018	0.00	1.000	-2.52447	2.52447

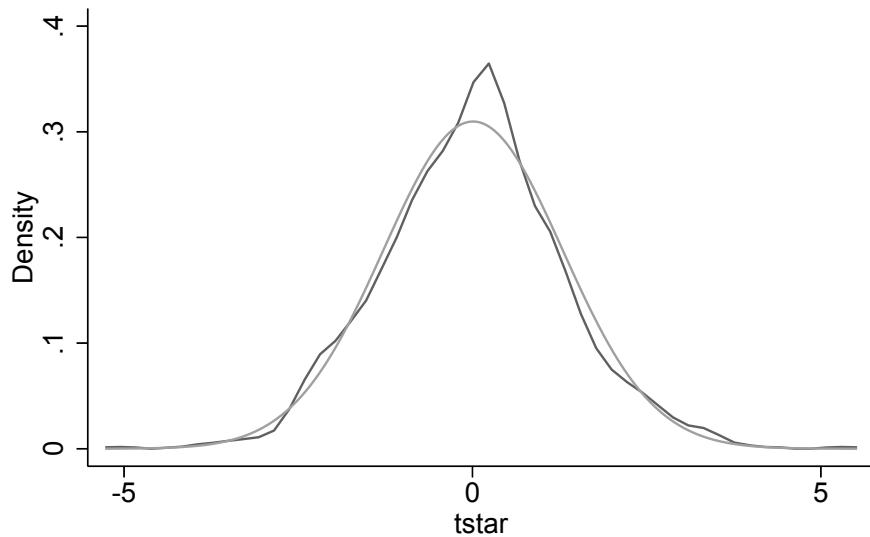
The output indicates that the distribution of  $t^*$  is considerably more dispersed than a standard normal, with a standard deviation of 1.29 rather than 1.0 for the standard normal.

The percentile- $t$  tests and confidence intervals for `chronic` are based on the 999 values of  $t^*$  saved in `percentilet.dta`.

```
. * percentile-t: Plot the density of tstar
. use percentilet, clear
(bootstrap: poisson)
. tabstat tstar, stats(count mean sd skew kurt)
Variable |       N        Mean         SD      Skewness    Kurtosis
tstar    |   999   .0086038   1.288018   .0729155   3.491272
.
. kdensity tstar, bw(0.2) normal legend(off) xtitle("tstar")
>      note(" ") scale(1.2) plotregion(style(none)) title(" ")
```

The  $t^*$  values have approximate mean of zero, little asymmetry, and little nonnormal kurtosis. The big departure from the asymptotic result of an  $N(0, 1)$  distribution is that the standard deviation of 1.29 is much greater than 1.

Figure 12.1 reveals that the density of  $t^*$  is symmetric but is a bit more peaked than the normal. And, as already noted, the standard deviation is considerably greater than the standard normal value of one. Thus, we expect the percentile- $t$  method to lead to larger  $p$ -values and wider confidence intervals.



**Figure 12.1.** Distribution of  $t^*$  from pairs percentile- $t$  bootstrap

The  $p$ -value for a symmetric two-sided Wald test of  $H_0 : \beta_{\text{chronic}} = 0$  is obtained as follows

```
. * Percentile-t p-value for symmetric two-sided Wald test of H0: theta = 0
. qui count if abs(`theta`/`setheta`) < abs(tstar)
. display "p-value = " r(N)/_N
p-value = .14614615
```

We do not reject  $H_0 : \beta_{\text{chronic}} = 0$  against  $H_0 : \beta_{\text{chronic}} \neq 0$  at the 0.05 level, because  $p = 0.146 > 0.05$ . By comparison, if we use the usual standard normal critical values,  $p = 0.056$ , which is considerably smaller.

The above code can be adapted to apply to several or all parameters by using the `bootstrap` prefix to obtain `_b` and `_se`, saving these in a file, using this file, and computing for each parameter of interest the values  $t^*$  given  $\hat{\theta}^*$ ,  $\hat{\theta}$ , and  $\text{se}(\hat{\theta}^*)$ .

### 12.5.3 Percentile-t Wald confidence interval

We obtain a percentile- $t$  95% confidence interval for the coefficient of `chronic` using the simpler (12.6), rather than inverting the test defined in

(12.5), where  $t_1^*, \dots, t_B^*$  were obtained in the previous section. We have

```
. * Percentile-t confidence interval
. _pctile tstar, p(2.5,97.5)
. scalar lb = `theta' + r(r1)*`settheta'
. scalar ub = `theta' + r(r2)*`settheta'

. display "2.5 and 97.5 percentiles of t* distn: " r(r1) ", " r(r2) _n
>      "95 percent percentile-t confidence interval is (" lb "," ub ")"
2.5 and 97.5 percentiles of t* distn: -2.4142661, 2.6618268
95 percent percentile-t confidence interval is (-.26122706,2.3554449)
```

The confidence interval is  $[-0.26, 2.36]$ , compared with  $[-0.03, 1.99]$ , which is obtained by using the robust estimate of the VCE with `poisson`. The wider confidence interval is due to the bootstrap- $t$  critical values of  $-2.41$  and  $2.66$ , much larger than the standard normal critical values of  $-1.96$  and  $1.96$ . The confidence interval is also wider than the other bootstrap confidence intervals given in section [12.3.8](#).

Percentile- $t$  95% confidence intervals, like BCA confidence intervals, have the advantage of having a coverage rate of  $0.95 + O(N^{-1})$  rather than  $0.95 + O(N^{-1/2})$ . [Efron and Tibshirani \(1993\)](#), 160, 184, 188, 326) favor the BCA method for confidence intervals because it is transformation-respecting and percentile- $t$  can perform erratically in small-sample nonparametric settings. But they state on page 326 that “generally speaking, the bootstrap- $t$  works well for location parameters,” and regression coefficients are location parameters. Econometrics theoretical and empirical studies use the percentile- $t$  method if asymptotic refinement is desired.

## 12.6 Wild bootstrap with asymptotic refinement

The wild bootstrap is a bootstrap with asymptotic refinement (using percentile- $t$  methods) that uses the wild bootstrap resampling method, rather than the pairs resampling method used in the preceding section. Monte Carlo studies find that the wild bootstrap generally performs better than the pairs bootstrap. The wild bootstrap can be easily implemented for many common estimators using the `boottest` command presented below in section [12.6.2](#). And the `didregress` and `xtdidregress` commands include the wild cluster bootstrap as an option.

The wild bootstrap was first proposed in econometrics for OLS with heteroskedastic errors. See [Horowitz \(2001, 3215–3217\)](#), [Davison and Hinkley \(1997, 272\)](#), or [Cameron and Trivedi \(2005, 376\)](#) for discussion. There were few applications in this case because if  $N$  is so small that standard asymptotic methods provide a poor guide, then often there is no point in continuing analysis: the small sample size leads to very imprecise parameter estimates.

Subsequently, the wild bootstrap has been extended to two common settings in applied microeconomics studies where the precision of estimated coefficients may be reasonable yet standard asymptotic methods can perform poorly.

[Cameron, Gelbach, and Miller \(2008\)](#) proposed cluster-robust inference for OLS with clustered data or short panel data with few clusters. Given sufficient observations per cluster, estimation can be quite precise, but the usual asymptotic methods based on  $G \rightarrow \infty$ , where  $G$  is the number of clusters, performs very poorly when  $G$  is small; see section [6.4.6](#).

[Davidson and MacKinnon \(2010\)](#) considered inference for 2SLS with heteroskedastic errors when instruments are weak, in which case asymptotic theory can perform poorly even for quite large  $N$ ; see section [7.5](#).

The preceding wild bootstraps require resampling residuals, limiting analysis to least-squares estimators in models with additive errors. [Kline and Santos \(2012\)](#) proposed resampling the score, rather than the residuals. This

extends the methods to  $m$ -estimators, such as the logit MLE and nonlinear generalized method of moments estimators. They considered both heteroskedastic–robust and cluster–robust inference.

The discussion below considers all of these examples of the wild bootstrap, with focus on the wild cluster bootstrap for OLS.

Before doing so, we note that developing better inferential methods when the usual asymptotic methods perform poorly in finite samples, with incorrectly sized tests and confidence intervals with incorrect coverage rates, is an active area of research. One approach is to develop new methods in specific contexts, such as those for few clusters mentioned in section 6.4.6 and for weak instruments presented in section 7.7. An alternative approach is to use bootstraps with asymptotic refinement. These new methods are proven mathematically to be asymptotically better than existing methods and hence hopefully perform better in finite samples of typical size. But there is no guarantee. Finite-sample performance is established by Monte Carlo experiments, with designs that mimic typical settings encountered in practice. This improved performance in Monte Carlo experiments does not necessarily carry over to a specific application with real data.

### 12.6.1 Wild cluster bootstrap

We consider inference with clustered errors and few clusters in the linear model  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$ . For simplicity, we test a restriction on a single parameter, with null hypothesis  $H_0 : \beta_j = \beta_{j0}$ . Inference is based on the Wald test statistic  $t = (\hat{\beta}_j - \beta_{j0})/\text{se}(\hat{\beta}_j)$ , where  $\text{se}(\hat{\beta}_j)$  is a cluster–robust standard error and  $\hat{\boldsymbol{\beta}}$  is obtained by OLS regression of  $\mathbf{y}$  on  $\mathbf{X}$ . A pairs cluster bootstrap resamples over  $(y_g, \mathbf{x}_g)$ , where  $g$  denotes the  $g$ th of  $G$  clusters.

We instead use a wild cluster bootstrap. The benchmark method proceeds as follows; a number of variants are discussed later.

First, obtain residuals  $\hat{\mathbf{u}}_g = \mathbf{y}_g - \mathbf{X}_g \hat{\boldsymbol{\beta}}_{\text{rest}}$ , for the  $g$ th cluster, where  $\hat{\boldsymbol{\beta}}_{\text{rest}}$  is the restricted estimate obtained by OLS regression of  $\mathbf{y}$  on  $\mathbf{X}$  that imposes  $H_0$ . For the test of  $H_0 : \beta_j = 0$ , this just drops the  $j$ th regressor from the equation.

Second, in the  $b$ th of  $B$  bootstraps, a) for cluster  $g$  of  $G$  clusters, randomly set  $\hat{\mathbf{u}}_g^* = \hat{\mathbf{u}}_g$  with probability 0.5 or  $\hat{\mathbf{u}}_g^* = -\hat{\mathbf{u}}_g$ ; b) form new outcome variables  $\mathbf{y}_g^* = \mathbf{X}_g \hat{\boldsymbol{\beta}}_{\text{rest}} + \hat{\mathbf{u}}_g^*$ ; c) regress the resulting  $\mathbf{y}^*$  on  $\mathbf{X}$  giving estimate  $\hat{\boldsymbol{\beta}}^*$  and hence  $\hat{\beta}_j^*$  with cluster-robust standard error  $\text{se}(\hat{\beta}_j^*)$ ; and d) calculate  $t^* = (\hat{\beta}_j^* - \hat{\beta}_j)/\text{se}(\hat{\beta}_j^*)$ .

Third, obtain asymptotic refinement by applying a percentile- $t$  method given in section [12.5.1](#) to the  $B$  Wald statistics  $t_1^*, \dots, t_B^*$ .

This bootstrap is wild in that it chooses only one of two possible values for  $\hat{\mathbf{u}}_g^*$  and hence  $\mathbf{y}_g^*$ . Doing so over  $G$  clusters yields  $2^G$  possible realizations of the data  $(\mathbf{y}, \mathbf{X})$ . The weights on the residuals of either  $-1$  or  $1$  are called Rademacher weights. [Canay, Santos, and Shaikh \(2021\)](#) provide an alternative derivation as a randomization test with  $G$  fixed and the number of observations per cluster going to infinity. In that case, the clusters must satisfy a strong homogeneity restriction.

## 12.6.2 The bootttest command

The community-contributed `bootttest` command ([Roodman et al. 2019](#)) is a postestimation command that can be used following linear OLS and IV regression, including `regress` and `ivregress`, and nonlinear regression, including `logit`, `glm`, and `gsem`. It can also follow linear regression commands such as `areg`, `xtreg`, `fe`, and `xtivreg`, `fe` that absorb a single set of fixed effects.

The command format is

```
bootttest [indepelist] [, options]
```

where *indepelist* is a list of hypotheses to be tested separately. In the simplest case, the test is on a single regressor, for a test that a single regressor's coefficient is zero, or a particular value such as `bootttest x=0.5`.

The `weighttype()` option determines the way that the errors  $\hat{\mathbf{u}}_g^*$  are derived from the original errors  $\hat{\mathbf{u}}_g$ . The default uses Rademacher weights that randomly set  $\hat{\mathbf{u}}_g^* = \hat{\mathbf{u}}_g$  with probability 0.5 or  $\hat{\mathbf{u}}_g^* = -\hat{\mathbf{u}}_g$  with

probability 0.5. The new residuals have the same mean and variance as the original residuals. Mammen and gamma weights additionally match the third moment of the residuals, so they are potentially better with skewed residuals. But, unlike Rademacher weights, they do not match the fourth moments of the residuals, and simulations suggest that matching the fourth moment is more important. Normal weights multiply  $\hat{\mathbf{u}}_g$  by a draw from the standard normal distribution.

The two-point Rademacher weights lead to at most  $2^G$  bootstrap samples. When  $G < 10$ , it is better to use the `weighttype(webb)` option that uses 6-point Webb weights that randomly set  $\hat{\mathbf{u}}_g^* = a_g \times \hat{\mathbf{u}}_g$ , where  $a_g$  takes 1 of the 6 weights  $\pm \sqrt{3/2}$ ,  $\pm 1$ ,  $\pm \sqrt{1/2}$  with equal probabilities of 1/6.

There are many other options; only a few are mentioned here. The `seed(#)` option should always be used for replicability. The `reps(#)` option has a default of 999. The larger the value, the less results vary with the choice of seed, and more generally set  $B$  such that  $(B + 1)/\alpha$  is an integer. The default is to use a version of the wild bootstrap that is appropriate given the specified `vce()` for the estimator. The `robust` and `cluster()` options override these defaults. The `nonull` option does not impose the null hypothesis before bootstrapping, using  $\mathbf{y}_g^* = \mathbf{X}_g \hat{\boldsymbol{\beta}} + \hat{\mathbf{u}}_g^*$  rather than  $\mathbf{y}_g^* = \mathbf{X}_g \hat{\boldsymbol{\beta}}_{\text{rest}} + \hat{\mathbf{u}}_g^*$ . This is also theoretically justified, but simulations find it is better to impose the null; intuitively, efficiency is improved by imposing the null hypothesis. The `pvalue()` option sets the  $p$ -value type to `symmetric` (the default), `equaltail` [see (12.4)], `lower`, and `upper`.

In addition to the Wald test, the `boottest` command can be applied to the score (or Lagrange multiplier) test and, for IV estimation, the Anderson–Rubin (AR) test. The latter use of `boottest` enables cluster–robust inference with few clusters when instruments are weak. Finally, the option `score` provides the alternative wild score bootstrap, which can be used for nonlinear model estimators such as the logit MLE.

### 12.6.3 Wild cluster bootstrap example

As a few clusters example, we use a dataset from the U.S. Current Population Survey based on the study by [Hersch \(1998\)](#). Interest lies in how

individual log hourly wage (`lnw`) varies with the job injury rate in the individual's occupation (`occrate`). The other regressors are potential years of work experience and its square (`potexp`, `potexpsq`), years of education (`educ`), union membership (`union`), race (`nonwhite`), and region dummies (`northe`, `midw`, `west`).

There are two complications. First, `occrate` is the same for all individuals in that occupation, so we need to obtain standard errors that are clustered on the occupation identifier `occ_id`. Second, we use an extract that covers only 10 occupations, so  $G = 10$ , and there is a few clusters problem.

We first perform OLS with heteroskedastic–robust standard errors that ignore clustering.

```
. * Few clusters OLS: Heteroskedastic-robust standard errors
. qui use mus212occfewcluster, clear
. global xlist potexp potexpsq educ union nonwhite northe midw west
. regress lnw occrate $xlist, vce(robust)

Linear regression                               Number of obs      =      1,594
                                                F(9, 1584)        =      90.00
                                                Prob > F        =     0.0000
                                                R-squared        =     0.3589
                                                Root MSE         =     .45176
```

lnw	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
occrate	-.0283627	.003188	-8.90	0.000	-.0346158	-.0221095
potexp	.0400968	.0037081	10.81	0.000	.0328235	.0473701
potexpsq	-.0005958	.0000843	-7.07	0.000	-.0007612	-.0004304
educ	.0870259	.0062229	13.98	0.000	.0748199	.0992318
union	.2246745	.0310659	7.23	0.000	.1637398	.2856092
nonwhite	-.08293	.0377609	-2.20	0.028	-.1569966	-.0088634
northe	.0462904	.0335869	1.38	0.168	-.0195892	.1121699
midw	-.0175544	.0322813	-0.54	0.587	-.0808729	.045764
west	.0371981	.0338121	1.10	0.271	-.0291231	.1035193
_cons	.9046225	.0957027	9.45	0.000	.7169052	1.09234

Surprisingly, wages fall with job injury risk, even after controlling for regressors that appear to be mostly statistically significant ([Hersch \[1998\]](#) obtained the expected positive sign when attention is restricted to female workers).

We then perform OLS with cluster-robust standard errors. The `regress` command bases inference based on the  $t(9)$  distribution because  $G - 1 = 9$ .

```
. * Few clusters OLS: Cluster-robust standard errors
. regress lnw occrate $xlist, vce(cluster occ_id)

Linear regression
Number of obs      =      1,594
F(8, 9)            =
Prob > F          =
R-squared          =      0.3589
Root MSE           =     .45176

(Std. err. adjusted for 10 clusters in occ_id)
```

lnw	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
occrate	-.0283627	.0104466	-2.72	0.024	-.0519945	-.0047308
potexp	.0400968	.007381	5.43	0.000	.0233998	.0567938
potexpsq	-.0005958	.0001146	-5.20	0.001	-.000855	-.0003366
educ	.0870259	.017553	4.96	0.001	.0473183	.1267334
union	.2246745	.0887073	2.53	0.032	.0240046	.4253444
nonwhite	-.08293	.0538803	-1.54	0.158	-.2048157	.0389558
northe	.0462904	.0220224	2.10	0.065	-.0035277	.0961085
midw	-.0175544	.0281175	-0.62	0.548	-.0811605	.0460517
west	.0371981	.0348024	1.07	0.313	-.0415303	.1159265
_cons	.9046225	.2433333	3.72	0.005	.3541643	1.455081

The cluster-robust standard error of the cluster-invariant regressor `occrate` is 3.28 times larger. This larger standard error combined with the use of the  $t(9)$  distribution results in a much larger  $p = 0.024$ . Note that the overall  $F$  statistic is not reported. This is because the rank of the cluster-robust VCE can be shown to be at most the minimum of the number of regressor ( $K$ ) and  $G - 1$ , and here  $K = 10$  and  $G - 1 = 9$ . We could nonetheless perform, for example, a test of eight linearly dependent restrictions.

Simulation studies find that even using  $t(G - 1)$  critical values leads to overrejection. This is especially the case if the clusters are unbalanced because of differences in cluster sizes or substantial differences in regressor values across clusters. The community-contributed `clusteff` command ([Lee and Steigerwald 2018](#)), based on results in [Carter, Schnepel, and Steigerwald \(2017\)](#), provides a conservative estimate of the effective number of clusters.

Applying the `clusteff` command yields

```
. * Few clusters OLS: CSS conservative effective number of clusters
. clusteff lnw occrate $covars, cluster(occ_id) test(occrate)
Number of clusters: 10
Estimated effective number of clusters: 4.6429332
Warning: G* estimated to be below 50.
```

The estimated effective number of clusters  $G^*$  equals 4.64. The estimate is conservative because it is based on the assumption of perfect intracluster correlation of the errors. If we use the ad hoc adjustment of  $G^* - 1$  degrees of freedom, a  $t(4)$  test yields  $p = 0.053 (=2*ttail(4, 2.72))$  rather than  $p = 0.024$  using the  $t(9)$  distribution.

We apply the `boottest` postestimation command with default settings, aside from the following adjustments. First, for replicability we set the seed. Second, to reduce dependence on the seed value, we set the number of simulations to 9,999, rather than the default 999; computation following OLS is nonetheless very fast. Third, because with  $G = 10$  the default Rademacher weights lead to only  $2^{10} = 1024$  possible bootstrap samples, we use Webb weights. We obtain

```
. * Few clusters OLS: Wild cluster bootstrap with Webb weights
. boottest occrate, seed(10101) reps(9999) weight(webb)
Wild bootstrap-t, null imposed, 9999 replications, Wald test, bootstrap
> clustering by occ_id, Webb weights:
    occrate
        t(9) =      -2.7150
        Prob>|t| =      0.0230
95% confidence set for null hypothesis expression: [-.07838, -.01119]
```

The test is one of whether the coefficient of `occrate` equals zero. Using the equal-tail two-sided test defined in (12.5) yields  $p = 0.023$ , so we reject  $H_0$  at level 0.05 and conclude that `occrate` is statistically significant. The corresponding confidence interval, found by inverting equal-tail two-sided tests, is  $[-0.078, -0.011]$ .

#### 12.6.4 Score-based wild cluster bootstrap

The wild cluster bootstrap described above resamples the residuals. [Kline and Santos \(2012\)](#) proposed instead resampling the score, which is the first derivative of the objective function and for OLS equals  $\mathbf{X}'\mathbf{u}$ . The advantage

of resampling the score is that we can extend the method to nonlinear models, such as an  $m$ -estimator that maximizes objective function  $Q(\boldsymbol{\theta})$  has score  $\partial Q(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$ .

Further details are given in [Kline and Santos \(2012\)](#). To give some intuition for the method, we consider linear regression. The wild bootstrap sets  $\mathbf{y}^* = \mathbf{X}\widehat{\boldsymbol{\beta}}_{\text{rest}} + \widehat{\mathbf{u}}^*$ , so  $\widehat{\boldsymbol{\beta}}^* = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}^* = \widehat{\boldsymbol{\beta}}_{\text{rest}} + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\widehat{\mathbf{u}}^*$ . It follows that in the  $b$ th resample,

$$\begin{aligned}\widehat{\boldsymbol{\beta}}^* &= \widehat{\boldsymbol{\beta}}_{\text{rest}} + (\mathbf{X}'\mathbf{X})^{-1} \sum_g \mathbf{X}_g \widehat{\mathbf{u}}_g^* \\ &= \widehat{\boldsymbol{\beta}}_{\text{rest}} + (\mathbf{X}'\mathbf{X})^{-1} \sum_g \mathbf{X}_g (a_g \widehat{\mathbf{u}}_g) \\ &= \widehat{\boldsymbol{\beta}}_{\text{rest}} + (\mathbf{X}'\mathbf{X})^{-1} \sum_g (a_g \mathbf{X}_g \widehat{\mathbf{u}}_g) \\ &= \widehat{\boldsymbol{\beta}}_{\text{rest}} + (\mathbf{X}'\mathbf{X})^{-1} \sum_g (\mathbf{X}_g \widehat{\mathbf{u}}_g)^*\end{aligned}$$

where  $(\mathbf{X}_g \widehat{\mathbf{u}}_g)^* = a_g \mathbf{X}_g \widehat{\mathbf{u}}_g$  with  $a_g$  the wild bootstrap weights such as Rademacher weights. So we could obtain the same estimate  $\widehat{\boldsymbol{\beta}}^*$  by randomly weighting  $\mathbf{X}_g \widehat{\mathbf{u}}_g$ , rather than randomly weighting  $\widehat{\mathbf{u}}_g$  and multiplying by  $\mathbf{X}_g$ . For example, randomly set  $\mathbf{X}_g \widehat{\mathbf{u}}_g^* = \mathbf{X}_g \widehat{\mathbf{u}}_g$  with probability 0.5 or  $\mathbf{X}_g \widehat{\mathbf{u}}_g^* = -\mathbf{X}_g \widehat{\mathbf{u}}_g$  with probability 0.5.

We illustrate this in a logit model example. The dependent variable equals 1 if the hourly wage exceeds 10 and equals 0 otherwise, and the number of clusters is reduced to 7. The usual logit estimates follow:

```

. * Few clusters logit: Cluster-robust standard errors
. drop if occ_id==63 | occ_id==113 | occ_id==133
(461 observations deleted)
. generate dlnw = lnw > ln(10)
. logit dlnw occrate $xlist, nolog vce(cluster occ_id)

Logistic regression                                         Number of obs = 1,133
                                                               Wald chi2(5) = .
                                                               Prob > chi2 = .
Log pseudolikelihood = -585.41192                         Pseudo R2 = 0.2531
                                                               (Std. err. adjusted for 7 clusters in occ_id)

```

dlnw	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
occrate	-.1227231	.0574737	-2.14	0.033	-.2353694 -.0100768
potexp	.1907595	.0403517	4.73	0.000	.1116716 .2698474
potexpsq	-.003126	.0006825	-4.58	0.000	-.0044637 -.0017883
educ	.3215529	.1051269	3.06	0.002	.115508 .5275978
union	1.640939	.3621219	4.53	0.000	.9311933 2.350685
nonwhite	-.3351905	.215012	-1.56	0.119	-.7566063 .0862253
northe	.2244681	.120476	1.86	0.062	-.0116604 .4605967
midw	.1987152	.1778251	1.12	0.264	-.1498155 .5472459
west	.3509475	.2354718	1.49	0.136	-.1105689 .8124638
_cons	-5.860125	1.386332	-4.23	0.000	-8.577287 -3.142964

For nonlinear estimators such as logit, Stata computes *p*-values using the standard normal distribution. It is better to at least use the  $t(G - 1)$ , which for `occrate` yields  $p = 0.076$  (as  $2 * \text{ttail}(6, 2.14) = 0.076$ ) or from command `test occrate, df(6)`.

With only 7 clusters, there are at most  $2^7 = 128$  possible different bootstrap samples if the two-point Rademacher weights are used. So we use the six-point Webb weights. We obtain

```

. * Few clusters logit: Score wild cluster bootstrap with Webb weights
. boottest occrate, seed(10101) weight(webb) reps(999)
Re-running regression with null imposed.

Logistic regression                                         Number of obs = 1,133
Log likelihood = -606.96732                               Wald chi2(8)  = 229.15
                                                        Prob > chi2 = 0.0000
( 1) [dlnw]occrate = 0


```

dlnw	Coefficient	Std. err.	z	P> z	[95% conf. interval]
occrate	0 (omitted)				
potexp	.1924966	.0244205	7.88	0.000	.1446334 .2403598
potexpsq	-.0030514	.0005195	-5.87	0.000	-.0040696 -.0020333
educ	.3952463	.0371796	10.63	0.000	.3223756 .468117
union	1.310985	.1950676	6.72	0.000	.9286596 1.693311
nonwhite	-.3631165	.2221503	-1.63	0.102	-.798523 .0722901
northe	.3039718	.1968254	1.54	0.122	-.0817988 .6897425
midw	.1495338	.1932623	0.77	0.439	-.2292533 .5283209
west	.4067953	.20723	1.96	0.050	.000632 .8129586
_cons	-7.462204	.5626493	-13.26	0.000	-8.564976 -6.359431

Score bootstrap-t, null imposed, 999 replications, Wald test, bootstrap  
> clustering by occ\_id, Webb weights:

```

occrate
      z =      -2.2957
      Prob>|z| =      0.1221

```

The wild cluster bootstrap *p*-value for `occrate` is 0.122 rather than 0.033, using standard normal *p*-values, or 0.076, using  $t(G - 1)$  *p*-values for this example with  $G = 7$ . We note that a  $t(3)$  distribution yields  $p = 0.122$  because  $2*ttail(3, 2.14) = 0.1218$ .

## 12.6.5 Wild bootstrap for IV estimation

[Davidson and MacKinnon \(2010\)](#) propose use of the wild bootstrap following IV estimation for independent heteroskedastic errors. The `boottest` command implements this case as well as extension to clustered errors.

We apply the wild bootstrap to the data used in section [7.7](#), restricting analysis to the first 1,500 observations because computational time is much longer than following OLS. There is one endogenous regressor (`hi_empunion`) and two instruments (`ssiratio` and `firmsz`). The instruments

are reasonably strong so that in this example, the wild bootstrap should lead to little change.

Using heteroskedastic-robust standard errors, we obtain

```
. * Weak IV with independent observations: Wild bootstrap of Wald test
. qui use mus207mepspresdrugs, clear
. keep if _n <= 1500 & linc != .
(8,921 observations deleted)
. global x2list totchr age female blhisp linc
. ivregress 2sls ldrugexp $x2list (hi_empunion = ssiratio firmsz),
> noheader vce(robust)
```

ldrugexp	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
hi_empunion	-.5693183	.4678496	-1.22	0.224	-1.486287	.34765
totchr	.4658999	.0272349	17.11	0.000	.4125205	.5192792
age	-.0033616	.0055211	-0.61	0.543	-.0141828	.0074596
female	-.06974	.0791641	-0.88	0.378	-.2248989	.0854188
blhisp	-.1410898	.1049148	-1.34	0.179	-.346719	.0645395
linc	.0680306	.0488175	1.39	0.163	-.0276499	.1637111
_cons	5.606257	.518471	10.81	0.000	4.590073	6.622442

```
Instrumented: hi_empunion
Instruments: totchr age female blhisp linc ssiratio firmsz
. boottest hi_empunion, seed(101010) reps(999)
.....
Wild bootstrap-t, null imposed, 999 replications, Wald test, Rademacher weights:
hi_empunion
      z =      -1.2169
      Prob>|z| =      0.2372
95% confidence set for null hypothesis expression: [-1.541, .3135]
```

For the endogenous regressor `hi_empunion`, the wild bootstrap  $p = 0.237$  is close to 0.224 obtained using the usual asymptotics.

The AR test is specifically designed to accommodate weak instruments; see section [7.7.2](#). A wild bootstrap of the AR test yields the following.

```

. * Weak IV with independent observations: Wild bootstrap of AR test
. boottest, ar reps(999)

Wild bootstrap-t, null imposed, 999 replications, Anderson-Rubin Wald test,
> Rademacher weights:
hi_empunion

          chi2(2) =      2.2357
          Prob > chi2 =    0.3524

95% confidence set for null hypothesis expression: [-1.868, .4822]
. qui regress ldrugexp ssiratio firmsz $x2list, vce(robust)
. test ssiratio firmsz // The standard AR test for this example
( 1)  ssiratio = 0
( 2)  firmsz = 0
      F(  2,  1438) =     1.11
      Prob > F =    0.3293

```

The wild bootstrap  $p = 0.352$  is close to 0.329 obtained using the usual asymptotics for the AR test.

## 12.7 Bootstrap pairs using `bsample` and `simulate`

The `bootstrap` prefix can be used only if one can provide a single expression for the quantity being bootstrapped. If this is not possible, one can use the `bsample` command to obtain one bootstrap sample, compute the statistic of interest for this resample, and use the `simulate` or `postfile` command to execute this command a number of times.

### 12.7.1 The `bsample` command

The `bsample` command draws random samples with replacement from the current data in memory. The command syntax is

```
bsample [ exp ] [ if ] [ in ] [ , options ]
```

where *exp* specifies the size of the bootstrap sample, which must be at most the size of the selected sample. The `strata(varlist)`, `cluster(varlist)`, `idcluster(newvar)`, and `weight(varname)` options allow stratification, clustering, and weighting. The `idcluster()` option is discussed in section [12.3.5](#).

In using the `bsample` command, one should first set the seed for reproducibility.

### 12.7.2 The `bsample` command with `simulate`

As an example, we use the `bsample` and `simulate` commands to reproduce the percentile-*t* Wald test given in section [12.5.2](#). We first define the program for one bootstrap replication. The `bsample` command without argument produces one resample of all variables with a replacement of size *N* from the original sample of size *N*.

The program returns a scalar, `tstar`, that equals  $t^*$  in (12.3). Because we are not returning parameter estimates, we use an r-class program. We have

```

* Program to do one bootstrap replication
program onebootrep, rclass
    version 17
    drop _all
    use mus212bootdata
    bsample
    poisson docvis chronic, vce(robust)
    return scalar tstar = (_b[chronic]-$theta)/_se[chronic]
end

```

Note that robust standard errors are obtained here. The referenced global macro, `theta`, constructed below, is the estimated coefficient of `chronic` in the original sample. We could alternatively pass this as a program argument rather than use a global macro. The program returns `tstar`.

We next obtain the original sample parameter estimate and use the `simulate` prefix to run the `onebootrep` program  $B$  times. We have

```

. * Now do 999 bootstrap replications
. qui use mus212bootdata, clear
. qui poisson docvis chronic, vce(robust)
. global theta = _b[chronic]
. global settheta = _se[chronic]
. simulate tstar=r(tstar), seed(10101) reps(999) nodots
>     saving(percentile2, replace): onebootrep
        Command: onebootrep
        tstar: r(tstar)
(file percentile2.dta not found)

```

`percentile2.dta` has the 999 bootstrap values  $t_1^*, \dots, t_{999}^*$  that can then be used to calculate the bootstrap  $p$ -value.

```

. * Analyze the results to get the p-value
. use percentile2, clear
(simulate: onebootrep)
. qui count if abs($theta/$settheta) < abs(tstar)
. display "p-value = " r(N)/_N
p-value = .14614615

```

The  $p$ -value is 0.146, leading to nonrejection of  $H_0: \beta_{\text{chronic}} = 0$  at the 0.05 level. This result is exactly the same as that in section [12.5.2](#) obtained using the `bootstrap` command.

Additional examples of the use of the `bsample` command are given in section [12.8](#).

## 12.8 Alternative resampling schemes

There are many ways to resample other than the nonparametric pairs and cluster-pairs bootstraps methods used by the Stata bootstrap commands. These other methods can be performed by using an approach similar to the one in section [12.7.2](#), with a program written to obtain one bootstrap resample and calculate the statistics of interest, and this program then called  $B$  times. We do so for several methods, bootstrapping regression model parameter estimates.

The programs are easily adapted to bootstrapping other quantities, such as the  $t$  statistic to obtain asymptotic refinement. For asymptotic refinement, there is particular benefit in using methods that exploit more information about the DGP than is used by bootstrap pairs. This additional information includes holding  $\mathbf{x}$  fixed through the bootstraps, called a design-based or model-based bootstrap; imposing conditions such as  $E(u|\mathbf{x}) = 0$  in the bootstrap; and for hypothesis tests, imposing the null hypothesis on the bootstrap resamples. See, for example, [Horowitz \(2001\)](#), [MacKinnon \(2002\)](#), and [Cameron, Gelbach, and Miller \(2008\)](#).

### 12.8.1 Bootstrap pairs resampling scheme

We begin with bootstrap pairs, repeating code similar to that in section [12.7.2](#). The following program obtains one bootstrap resample by resampling from the original data with replacement.

```
* Program to resample using bootstrap pairs
program bootpairs
    version 17
    drop _all
    use mus212bootdata
    bsample
    poisson docvis chronic
end
```

To check the program, we run it once.

```
. * Check the program by running once
. bootpairs
(output omitted)
```

We then run the program 400 times. We have

```
. * Bootstrap pairs for the parameters
. simulate _b, seed(10101) reps(400) nodots: bootpairs
    Command: bootpairs
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis_b_c^c	400	.9880522	.5386575	-.5826053	2.693661
docvis_b_c^s	400	.9602076	.3536507	-.0689929	1.747957

The bootstrap estimate of the standard error of  $\beta_{\text{chronic}}$  equals 0.5387, as in section [12.3.3](#).

## 12.8.2 Parametric bootstrap resampling scheme

A parametric bootstrap is essentially a Monte Carlo simulation. Typically, we hold  $\mathbf{x}_i$  fixed at the sample values; replace  $y_i$  by a random draw,  $y_i^*$ , from the density  $f(y_i|\mathbf{x}_i, \boldsymbol{\theta})$  with  $\boldsymbol{\theta}$  evaluated at the original sample estimate,  $\hat{\boldsymbol{\theta}}$ ; and regress  $y_i^*$  on  $\mathbf{x}_i$ . A parametric bootstrap requires much stronger assumptions, correct specification of the conditional density of  $y$  given  $\mathbf{x}$ , than the paired or nonparametric bootstrap.

To implement a parametric bootstrap, we adopt the preceding `bootpairs` program and replace the `bsample` command with code to randomly draw  $y$  from  $f(y|\mathbf{x}, \hat{\boldsymbol{\theta}})$ .

For doctor visits, which are overdispersed count data, we use the negative binomial distribution rather than the Poisson. We first obtain the negative binomial parameter estimates,  $\hat{\boldsymbol{\theta}}$ , using the original sample. In this case, it is sufficient and simpler to obtain the fitted mean,  $\hat{\mu}_i = \exp(\mathbf{x}'_i \hat{\boldsymbol{\beta}})$ , and the dispersion parameter,  $\hat{\alpha}$ . We have

```

. * Fit the model with original actual data and save estimates
. use mus212bootdata
. quietly nbreg docvis chronic
. predict muhat
. global alpha = e(alpha)

```

We use these estimates to obtain draws of  $y$  from the negative binomial distribution given  $\hat{\alpha}$  and  $\hat{\mu}_i$ , using a Poisson–gamma mixture (explained in section 20.2.2). The `rgamma(1/a, a)` function draws a gamma variable,  $\nu$ , named `nu` with a mean of 1 and a variance of  $a$ , and the `rpoisson(nu*mu)` function then generates negative binomial draws with a mean of  $\mu$  and a variance of  $\mu + a\mu^2$ . We have

```

* Program for parametric bootstrap generating from negative binomial
program bootparametric, eclass
    version 17
    capture drop nu dvhat
    generate nu = rgamma(1/$alpha,$alpha)
    generate dvhat = rpoisson(muhat*nu)
    nbreg dvhat chronic
end

```

We check the program by using the `bootparametric` command and then bootstrap 400 times.

```

. * Parametric bootstrap for the parameters
. simulate _b, seed(10101) reps(400) nodots: bootparametric
    Command: bootparametric

. summarize

```

Variable	Obs	Mean	Std. dev.	Min	Max
<code>dvhat_b_ch~c</code>	400	.9234596	.4640827	-.8095576	2.208154
<code>dvhat_b_cons</code>	400	.9980879	.2401054	.3483067	1.625967
<code>_eq2_b_lna~a</code>	400	.5019578	.2825218	-.4267736	1.298986

Because we generate data from a negative binomial model and we fit a negative binomial model, the average of the 400 bootstrap coefficient estimates should be close to the DGP values. This is the case here. Also, the bootstrap standard errors are within 10% of those from the negative binomial estimation of the original model, not given here, suggesting that the negative binomial model may be a reasonable one for these data.

### 12.8.3 Residual bootstrap resampling scheme

For linear OLS regression, under the strong assumption that errors are independent and identically distributed (i.i.d.), an alternative to bootstrap pairs is a residual bootstrap. This holds  $\mathbf{x}_i$  fixed at the sample values and replaces  $y_i$  with  $y_i^* = \mathbf{x}'_i \hat{\beta} + \hat{u}_i^*$ , where  $\hat{u}_i^*$  are bootstrap draws from the original sample residuals  $\hat{u}_1, \dots, \hat{u}_N$ . This bootstrap, sometimes called a design bootstrap, can lead to better performance of the bootstrap by holding regressors fixed.

The `bootpairs` program is adapted by replacing `bsample` with code to randomly construct  $\hat{u}_i^*$  from  $\hat{u}_i$  for each observation  $i$ . This is not straightforward, because the `bsample` command is intended to bootstrap the entire dataset in memory, whereas here we wish to bootstrap the residuals but not the regressors.

As illustration, we continue to use the `docvis` example, even though Poisson regression is more appropriate than OLS regression. The following code performs the residual bootstrap:

```
* Residual bootstrap for OLS with i.i.d. errors
use mus212bootdata, clear
quietly regress docvis chronic
predict uhat, resid
keep uhat
save residuals, replace
program bootresidual
    version 17
    drop _all
    use residuals
    bsample
    merge 1:1 _n using mus212bootdata
    regress docvis chronic
    predict xb
    generate ystar = xb + uhat
    regress ystar chronic
end
```

We check the program by using the `bootresidual` command and bootstrap 400 times.

```

. * Residual bootstrap for the parameters
. simulate _b, seed(10101) reps(400) nodots: bootresidual
    Command: bootresidual

. summarize



| Variable   | Obs | Mean     | Std. dev. | Min       | Max      |
|------------|-----|----------|-----------|-----------|----------|
| _b_chronic | 400 | 4.802995 | 2.296399  | -.9681436 | 12.29839 |
| _b_cons    | 400 | 2.766325 | 1.20739   | .2407407  | 6.493056 |


```

The output reports the average of the 400 slope coefficient estimates (4.803), close to the original sample OLS slope coefficient estimate, not reported, of 4.694. The bootstrap estimate of the standard error (2.30) is close to the original sample OLS default estimate, not given, of 2.39. This is expected because the residual bootstrap assumes that errors are i.i.d.

## 12.8.4 Wild bootstrap resampling scheme

The wild bootstrap was presented in section [12.6](#) and implemented using the community-contributed `boottest` command. Here we provide a simple example, without asymptotic refinement.

For linear regression with independent observations, a wild bootstrap accommodates the more realistic assumption that errors are independent but not identically distributed, permitting heteroskedasticity. This holds  $\mathbf{x}_i$  fixed at the sample values and replaces  $y_i$  with  $y_i^* = \mathbf{x}'_i \hat{\beta} + \hat{u}_i^*$ , where there are various ways to resample  $\hat{u}_i^*$ . Here we use Mammen weights that set  $\hat{u}_i^* = a_i \hat{u}_i$  and  $a_i = (1 - \sqrt{5})/2 \simeq -0.618$  with the probability  $(1 + \sqrt{5})/2\sqrt{5} \simeq 0.723$  and  $a_i = 1 - (1 - \sqrt{5})/2$  with the probability  $1 - (1 + \sqrt{5})/2\sqrt{5}$ . For each observation,  $\hat{u}_i^*$  takes only two possible values, but across all  $N$  observations, there are  $2^N$  possible resamples if the  $N$  values of  $\hat{u}_i$  are distinct.

The preceding `bootresidual` program is adapted by replacing `bsample` with code to randomly draw  $\hat{u}_i^*$  from  $\hat{u}_i$  and then form  $y_i^* = \mathbf{x}'_i \hat{\beta} + \hat{u}_i^*$ .

The Stata code is the same as that in section [12.8.1](#), except that the `bsample` command in the `bootpairs` program needs to be replaced with

code to randomly draw  $\hat{u}_i^*$  from  $\hat{u}_i$  and then form  $y_i^* = \mathbf{x}'_i \hat{\beta} + \hat{u}_i^*$ .

```
* Wild bootstrap for OLS with i.i.d. errors
use mus212bootdata, clear
program bootwild
    version 17
    drop _all
    use mus212bootdata
    regress docvis chronic
    predict xb
    predict u, resid
    gen ustard = -0.618034*u
    replace ustard = 1.618034*u if runiform() > 0.723607
    gen ystar = xb + ustard
    regress ystar chronic
end
```

We check the program by issuing the `bootwild` command and bootstrap 400 times.

```
. * Wild bootstrap for the parameters
. simulate _b, seed(10101) reps(400) nodots: bootwild
    Command: bootwild
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
_b_chronic	400	4.779422	2.986341	-2.205521	11.53673
_b_cons	400	2.807026	.957078	1.123328	5.709338

The wild bootstrap permits heteroskedastic errors and yields bootstrap estimates of the standard errors (2.99) that are close to the original sample OLS heteroskedasticity-robust estimates, not given, of 3.06. These standard errors are considerably higher than those obtained by using the residual bootstrap, which is clearly inappropriate in this example because of the inherent heteroskedasticity of count data.

The percentile- $t$  method with the wild bootstrap provides asymptotic refinement to Wald tests and confidence intervals in the linear model with heteroskedastic errors or clustered errors; see section [12.6](#).

## 12.8.5 Subsampling

The bootstrap fails in some settings, such as a nonsmooth estimator. Then a theoretically more robust resampling method is subsampling, which draws a resample that is considerably smaller than the original sample. [Politis, Romano, and Wolf \(1999\)](#) provide an introduction to this method.

The method is easily implemented using the `bsample` command. For example, to perform subsampling where the resamples have one-third as many observations as the original sample, replace the `bsample` command in the bootstrap pairs with `bsample int(_N/3)`, where the `int()` function truncates to an integer toward zero.

We caution against use of this method, however, because in practice it is very sensitive to the subsample size chosen.

## 12.9 The jackknife

The delete-one jackknife is a resampling scheme that forms  $N$  resamples of size  $(N - 1)$  by sequentially deleting each observation and then estimating  $\theta$  in each resample.

### 12.9.1 Jackknife method

Let  $\widehat{\theta}_i$  denote the parameter estimate from the sample with the  $i$ th observation deleted,  $i = 1, \dots, N$ , let  $\widehat{\theta}$  be the original sample estimate of  $\theta$ , and let  $\bar{\widehat{\theta}} = N^{-1} \sum_{i=1}^N \widehat{\theta}_i$  denote the average of the  $N$  jackknife estimates.

The jackknife has several uses. The BC jackknife estimate of  $\theta$  equals  $N\widehat{\theta} - (N - 1)\bar{\widehat{\theta}} = (1/N) \sum_{i=1}^N \{N\widehat{\theta} - (N - 1)\widehat{\theta}_i\}$ . The variance of the  $N$  pseudovalues  $\widehat{\theta}_i^* = N\widehat{\theta} - (N - 1)\widehat{\theta}_i$  can be used to estimate  $\text{Var}(\widehat{\theta})$ . The BCA method for a bootstrap with asymptotic refinement also uses the jackknife.

There are two variants of the jackknife estimate of the VCE. The Stata default is

$$\widehat{V}_{\text{Jack}}(\widehat{\theta}) = \left\{ \frac{1}{N(N-1)} \sum_{i=1}^N (\widehat{\theta}_i^* - \bar{\widehat{\theta}}) (\widehat{\theta}_i^* - \bar{\widehat{\theta}})' \right\}$$

and the `mse` option gives the variation

$$\widehat{V}_{\text{Jack}}(\widehat{\theta}) = \left\{ \frac{N-1}{N} \sum_{i=1}^N (\widehat{\theta}_i - \widehat{\theta}) (\widehat{\theta}_i - \widehat{\theta})' \right\}$$

The method entails  $N$  resamples, which requires considerable computation when  $N$  is large. The resamples are not random draws, so there is no seed to

set.

The use of the jackknife for estimation of the vce has been largely superseded by the bootstrap. In recent research, [Cattaneo, Jansson, and Ma \(2019\)](#) find that the jackknife can reduce bias in two-step estimators with many covariates at the first step and propose subsequent inference based on a percentile-*t* bootstrap of the jackknife-based BC estimate.

### 12.9.2 The vce(jackknife) option and the jackknife prefix

For many estimation commands, the `vce(jackknife)` option can be used to obtain the jackknife estimate of the VCE. For example,

Poisson regression						
		Jknife *				
docvis	Coefficient	std. err.	t	P> t	[95% conf. interval]	
chronic	.9833014	.6222999	1.58	0.121	-.2672571	2.23386
_cons	1.031602	.3921051	2.63	0.011	.2436369	1.819566

The jackknife estimate of the standard error of the coefficient of `chronic` is 0.62, larger than the value 0.54 obtained by using the `vce(bootstrap, reps(2000))` option and the value 0.52 obtained by using the `vce(robust)` option; see the `poisson` example in section [12.3.4](#).

The `jackknife` prefix operates similarly to `bootstrap`.

## 12.10 Additional resources

For many purposes, the `vce(bootstrap)` option of an estimation command suffices (see [R] *vce\_option*) possibly followed by `estat bootstrap`. For more advanced analysis, the `bootstrap` prefix and the `bsample` command can be used.

For applications that use more elaborate methods than those implemented with the `vce(bootstrap)` option, care is needed, and a good understanding of the bootstrap is recommended. References include Efron and Tibshirani (1993), Davison and Hinkley (1997), Horowitz (2001), Davidson and MacKinnon (2004, chap. 4), Cameron and Trivedi (2005, chap. 11), and Hansen (2022a, chap. 10). For bootstrap with asymptotic refinement using the wild bootstrap, see section 12.6 for references and description of the community-contributed `boottest` command.

## 12.11 Exercises

1. Use the same data as those created in section [12.3.3](#), except keep the first 100 observations and keep the variables `educ` and `age`. After a Poisson regression of `docvis` on an intercept and `educ`, give default standard errors, robust standard errors, and bootstrap standard errors based on 1,000 bootstraps and a seed of 10101.
2. For the Poisson regression in exercise 1, obtain the following 95% confidence intervals: normal-based, percentile, BC, and BCA. Compare these. Which, if any, is best?
3. Obtain a bootstrap estimate of the standard deviation of the estimated standard deviation of `docvis`.
4. Continuing with the regression in exercise 1, obtain a bootstrap estimate of the standard deviation of the robust standard error of  $\hat{\beta}_{\text{educ}}$ .
5. Continuing with the regression in exercise 1, use the percentile- $t$  method to perform a Wald test with asymptotic refinement of  $H_0: \beta = 0$  against  $H_a: \beta \neq 0$  at the 0.05 level, and obtain a percentile- $t$  95% confidence interval.
6. Use the data of section [12.3.3](#) with 50 observations. Give the command given at the end of this exercise. Use the data in `percentile.dta` to obtain the following for the coefficient of the `chronic` variable: 1) bootstrap standard error; 2) bootstrap estimate of bias; 3) normal-based 95% confidence interval; and 4) percentile- $t$  95% confidence interval. For the last, you can use the `centile` command. Compare your results with those obtained from `estat bootstrap, all` after a Poisson regression with the `vce(bootstrap)` option.

```
bootstrap bstar=_b[chronic], reps(999) seed(10101) nodots ///
    saving(percentile, replace): poisson docvis chronic
use percentile, clear
```

7. Continuing from the previous exercise, does the bootstrap estimate of the distribution of the coefficient of `chronic` appear to be normal? Use the `summarize` and `kdensity` commands.
8. Repeat the percentile- $t$  bootstrap at the start of section [12.5.2](#). Use `kdensity` to plot the bootstrap Wald statistics. Repeat for an estimation

by `poisson` with default standard errors, rather than `nbreg`. Comment on any differences.



# **Chapter 13**

## **Nonlinear regression methods**

## 13.1 Introduction

We now turn to nonlinear regression methods. In this chapter, we consider single-equation models fit using cross-sectional data with all regressors exogenous.

Compared with linear regression, nonlinear regression presents two complications. There is no explicit solution for the estimator, so computation of the estimator requires iterative numerical methods. And, unlike a linear model, the marginal effect (ME) of a change in a regressor is no longer simply the associated slope parameter. For standard nonlinear models, the first complication is easily handled. Simply changing the command from `regress y x` to `poisson y x`, for example, leads to nonlinear estimation and regression output that looks essentially the same as the output from `regress`. The second complication can often be dealt with by obtaining MES using the `margins` command.

In this chapter, we provide an overview of Stata's nonlinear estimation commands and subsequent calculation of standard errors, prediction, and computation of MES. The discussion is applicable for analysis after any Stata estimation command, including the commands listed in table 13.1. A complete list of estimation commands can be found using command `help estimation commands`, while `help contents_stat` groups estimation commands by category.

**Table 13.1.** Some estimation commands for linear and nonlinear cross-sectional models

Model type	Estimation command
Linear	<code>regress, cnsreg, areg, hetregress, ivregress, sem, eregress, etregress, qreg, boxcox, mvreg, sureg, reg3, mixed, xtreg, xtgl, xtrc, xtpcse, xtregar, xtivreg, xtaylor, xtabond</code>
Nonlinear least squares	<code>nl, nlsur, menl</code>
Binary	<code>logit, logistic, probit, cloglog, hetprobit, ivprobit, heckprobit, eprobit, eteffects, melogit, meprobit, mecloglog, xtlogit, xtprobit, xtcloglog, xteprobit, biprobit</code>
Multinomial	<code>mlogit, clogit, cmclogit, nlogit, cmmixlogit, mprobit, cmmprobit, mecloglog, slogit, cmxtmixlogit, xtlogit, xtprobit, xtmlogit</code>
Ordinal	<code>ologit, oprobit, hetoprobit, heckoprobit, eoprobit, meologit, meoprobit, cmroporbit, cmrologit</code>
Censored normal	<code>tobit, intreg, truncreg, ivtobit, eintreg, metobit, meintreg, xttobit, xtintreg, xtfrontier</code>
Selection normal	<code>etregress, eintreg, heckman, xteregress, xteintreg, xtheckman</code>
Durations	<code>stcox, stintcox, stcrreg, streg, stintreg, mestreg, xtstreg</code>
Counts	<code>poisson, nbreg, gnbreg, cpoisson, tpoisson, heckpoisson, tnbreg, zip, zinb, ivpoisson, etpoisson, mepoisson, menbreg, xtpoisson, xtnbreg</code>
Other nonlinear	<code>glm, gmm, meglm, gsem, xtgee, fmm, bayes</code>

The discussion of model-specific issues, particularly specification tests that are an integral part of the modeling cycle of estimation, specification testing, and reestimation, is presented in chapter 11 and in the model-specific chapters 17–22. Chapter 16 presents methods used to fit a nonlinear model, including methods used when no Stata command is available for that model.

## 13.2 Nonlinear example: Doctor visits

As a nonlinear estimation example, we consider Poisson regression to model count data on the number of doctor visits. There is no need to first read chapter 20 on count data because we provide any necessary background here.

Although the outcome is discrete, the only difference this makes is in the choice of log density. The `poisson` command is actually not restricted to counts and can be applied to any variable  $y \geq 0$ . The points made with the count-data example could equally well be made with, for example, logit or probit models for binary outcomes or the Weibull model for duration data.

### 13.2.1 Data description

We use the dataset from the 2002 Medical Expenditure Panel Survey, first used in chapter 10. We model the number of office-based physician visits (`docvis`) by persons in the United States aged 25–64 years. The sample excludes those receiving public insurance (Medicare and Medicaid) and is restricted to those working in the private sector but who are not self-employed.

The regressors used here are restricted to health insurance status (`private`), health status (`chronic`), and socioeconomic characteristics (`female` and `income`) to keep Stata output short. We have

```
. * Read in dataset, select one year of data, and describe key variables
. qui use mus210mepsdocvisyoung
. keep if year02==1
(25,712 observations deleted)
. describe docvis private chronic female income
```

Variable name	Storage type	Display format	Value label	Variable label
docvis	int	%8.0g		Number of doctor visits
private	byte	%8.0g		Private insurance
chronic	byte	%8.0g		Chronic condition
female	byte	%8.0g		Female
income	float	%9.0g		Income in \$ / 1000

We then summarize the data:

```
. * Summary of key variables
. summarize docvis private chronic female income
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	4,412	3.957389	7.947601	0	134
private	4,412	.7853581	.4106202	0	1
chronic	4,412	.3263826	.4689423	0	1
female	4,412	.4718948	.4992661	0	1
income	4,412	34.34018	29.03987	-49.999	280.777

The dependent variable is a nonnegative integer count, here ranging from 0 to 134. We see 33% of the sample have a chronic condition and 47% are female. We use the whole sample, including the three people who have negative income (obtained by using the `tabulate income` command).

The relative frequencies of `docvis`, obtained by using the `tabulate docvis` command, are 36%, 16%, 10%, 7%, and 5% for, respectively, 0, 1, 2, 3, and 4 visits, and 26% of the sample have 5 or more visits.

### 13.2.2 Poisson model description

The Poisson regression model specifies the count  $y$  to have a conditional mean of the exponential form

$$E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta}) \quad (13.1)$$

This ensures that the conditional mean is positive, which should be the case for any random variable that is restricted to be nonnegative. However, the key ME  $\partial E(y|\mathbf{x})/\partial x_j = \beta_j \exp(\mathbf{x}'\boldsymbol{\beta})$  now depends on both the parameter estimate  $\beta_j$  and the particular value of  $\mathbf{x}$  at which the ME is evaluated; see section [13.7](#).

The starting point for count analysis is the Poisson distribution, with the probability mass function  $f(y|\mathbf{x}) = e^{-\mu} \mu^y / y!$ . Substituting in  $\mu_i = \exp(\mathbf{x}'_i \boldsymbol{\beta})$  from (13.1) gives the conditional density for the  $i$ th observation. This in turn gives the log-likelihood function

$Q(\beta) = \sum_{i=1}^N \{-\exp(\mathbf{x}'_i \beta) + y_i \mathbf{x}'_i \beta - \ln y_i!\}$ , which is maximized by the maximum likelihood estimator (MLE). The Poisson MLE solves the associated first-order conditions that can be shown to be

$$\sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \beta)\} \mathbf{x}_i = \mathbf{0} \quad (13.2)$$

Equation (13.2) has no explicit solution for  $\beta$ . Instead,  $\hat{\beta}$  is obtained numerically by using methods explained in section 16.2.

What if the Poisson distribution is the wrong distribution for modeling doctor visits? In general, the MLE is inconsistent if the density is misspecified. However, the Poisson MLE requires only the much weaker condition that the conditional mean function given in (13.1) be correctly specified, because then the left-hand side of (13.2) has an expected value of zero. Under this weaker condition, robust standard errors rather than default maximum likelihood (ML) standard errors should be used; see section 13.4.5.

### 13.3 Nonlinear regression methods

We consider four classes of estimators: ML, nonlinear least squares (NLS), generalized linear models (GLM), and generalized method of moments (GMM).

The first three are examples of  $m$  estimators that maximize (or minimize) an objective function of the form

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^N q_i(y_i, \mathbf{x}_i, \boldsymbol{\theta}) \quad (13.3)$$

where  $y$  denotes the dependent variable,  $\mathbf{x}$  denotes regressors (assumed exogenous),  $\boldsymbol{\theta}$  denotes a parameter vector, and  $q(\cdot)$  is a specified scalar function that varies with the model and estimator. In the Poisson case,  $\boldsymbol{\beta} = \boldsymbol{\theta}$ ; more generally,  $\boldsymbol{\beta}$  is a component of  $\boldsymbol{\theta}$ . Separate treatment of GMM is given in section [13.3.9](#).

#### 13.3.1 MLE and quasi-MLE and robust standard errors

MLES maximize the log-likelihood function. For  $N$  independent observations, the MLE  $\widehat{\boldsymbol{\theta}}$  maximizes

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^N \ln f(y_i | \mathbf{x}_i, \boldsymbol{\theta})$$

where  $f(y|\mathbf{x}, \boldsymbol{\theta})$  is the conditional density, for continuous  $y$ , or the conditional probability mass function, for discrete  $y$ .

If the density  $f(y|\mathbf{x}, \boldsymbol{\theta})$  is correctly specified, then the MLE is the best estimator to use. It is consistent for  $\boldsymbol{\theta}$ , it is asymptotically normally

distributed, and it is fully efficient, meaning that no other estimator of  $\theta$  has a smaller asymptotic variance–covariance matrix of the estimator (VCE).

Of course, the true density is unknown. If  $f(y|\mathbf{x}, \theta)$  is incorrectly specified, then in general the MLE is inconsistent. It may then be better to use other methods that, while not as efficient as the MLE, are consistent under weaker assumptions than those necessary for the MLE to be consistent.

The MLE remains consistent even if the density is misspecified, however, provided that 1) the specified density is in the linear exponential family (LEF) and 2) the functional form for the conditional mean  $E(y|\mathbf{x})$  is correctly specified. The default estimate of the VCE of the MLE is then no longer correct, so we base the inference on a robust estimate of the VCE. Examples of the LEF are Poisson and negative binomial (with a known dispersion parameter) for count data, Bernoulli for binary data (including logit and probit), one-parameter gamma for duration data (including exponential), normal (with a known variance parameter) for continuous data, and the inverse Gaussian.

It follows that consistency of the MLE for many standard models, notably the normal, Poisson, logit, probit, exponential, and gamma, requires only that the conditional mean function be correctly specified. For most other models, however, any distributional misspecification leads to inconsistency of the MLE.

Furthermore, when the density is misspecified, default standard errors are incorrect, and robust standard errors should be used, regardless of whether the estimator remains consistent; see [White \(1982\)](#).

The term quasi-MLE, or pseudo-MLE, is used when estimation is by ML, but subsequent inference is done without assuming that the density is correctly specified. Throughout this book, we use robust standard errors, unless there is reason not to do so.

The `mlexp` command (see section [13.3.3](#)) and more general optimization commands presented in chapter 16 enable ML estimation for user-defined likelihood functions. For commonly used models, this is not necessary,

however, because specific Stata commands have been developed for specific models.

### 13.3.2 The poisson command

For Poisson, the ML estimator is obtained by using the `poisson` command. The syntax of the command is

```
poisson depvar [indepvars] [if] [in] [weight] [, options]
```

This syntax is the same as that for `regress`. The only relevant option for our analysis here is the `vce()` option for the type of estimate of the VCE.

The `poisson` command with the `vce(robust)` option yields the following results for the doctor-visits data. As already noted, to restrict Stata output, we use far fewer regressors than should be used to model doctor visits.

```
. * Poisson regression (command poisson)
. poisson docvis private chronic female income, vce(robust)

Iteration 0:  log pseudolikelihood = -18504.413
Iteration 1:  log pseudolikelihood = -18503.549
Iteration 2:  log pseudolikelihood = -18503.549

Poisson regression                                         Number of obs = 4,412
                                                               Wald chi2(4) = 594.72
                                                               Prob > chi2 = 0.0000
                                                               Pseudo R2 = 0.1930

Log pseudolikelihood = -18503.549
```

docvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
private	.7986652	.1090014	7.33	0.000	.5850263 1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167 1.201614
female	.4925481	.0585365	8.41	0.000	.3778187 .6072774
income	.003557	.0010825	3.29	0.001	.0014354 .0056787
_cons	-.2297262	.1108732	-2.07	0.038	-.4470338 -.0124186

The output begins with an iteration log because the estimator is obtained numerically by using an iterative procedure presented in sections 16.2 and 16.3. In this case, only two iterations are needed. Each iteration increases the log-likelihood function, as desired, and iterations cease when there was little change in the log-likelihood function. The term

“pseudolikelihood” is used rather than “log likelihood” because use of `vce(robust)` means that we no longer are maintaining that the data be exactly Poisson distributed. The remaining output from `poisson` is remarkably similar to that for `regress`.

Here the four regressors are jointly statistically significant at 5% because the `Wald chi2(4)` test statistic has  $p = 0.00 < 0.05$ . The pseudo- $R^2$  is discussed in section [13.8.1](#). There is no analysis-of-variance table, because this table is appropriate only for linear least squares with independent homoskedastic errors.

The remaining output indicates that all regressors are individually statistically significant at a level of 0.05 because all  $p$ -values are less than 0.05. For each regressor, the output presents the following in turn:

Coefficients	$\hat{\beta}_j$
Standard errors	$s_{\hat{\beta}_j}$
$z$ statistics	$z_j = \hat{\beta}_j / s_{\hat{\beta}_j}$
$p$ -values	$p_j = \Pr\{  z_j  > 0   z_j \sim N(0, 1) \}$
95% confidence intervals	$\hat{\beta}_j \pm 1.96 \times s_{\hat{\beta}_j}$

The  $z$  statistics and  $p$ -values are computed by using the standard normal distribution, rather than the  $t$  distribution with  $N - K$  degrees of freedom. The  $p$ -values are for a two-sided test of whether  $\beta_j = 0$ . For a one-sided test of  $H_0: \beta_j \leq 0$  against  $\beta_j > 0$ , the  $p$ -value is half of that reported in the table, provided that  $z_j > 0$ . For a one-sided test of  $H_0: \beta_j \geq 0$  against  $\beta_j < 0$ , the  $p$ -value is half of that reported in the table, provided that  $z_j < 0$ .

A nonlinear model raises a new issue of interpretation of the slope coefficients  $\beta_j$ . For example, what does the value 0.0036 for the coefficient of `income` mean? Given the exponential functional form for the conditional mean in [\(13.1\)](#), it means that a \$1,000 increase in income (a one-unit increase in `income`) leads to a 0.0036 proportionate increase, or a 0.36%

increase, in doctor visits. We address this important issue in detail in section [13.7](#).

Note that test statistics following nonlinear estimation commands such as `poisson` are based on the standard normal distribution and chi-squared distributions, whereas those following some linear estimation commands, notably the `regress` and `xtreg, fe` commands, use the  $t$  and  $F$  distributions. For independent observations, this makes little difference for larger samples, say,  $N > 100$ . For clustered observations with few clusters, this can make a big difference, even if there are many observations; see section [6.4.6](#). The postestimation command `test` with option `df()` uses the  $F$  distribution.

### 13.3.3 The `mlexp` command

Stata provides several optimization commands that enable ML estimation for a user-provided parametric model; see chapter 16. The simplest command is the `mlexp` command, suitable for independent observations with parameters entering through indexes such as  $\mathbf{x}_i'\boldsymbol{\beta}$ .

The command syntax is

```
mlexp (lexp) [if] [in] [weight] [, options]
```

where *lexp* is a substitutable expression (see section [13.3.6](#)) for the log density of a single observation. The only relevant option for our analysis here is the `vce()` option for the type of estimate of the VCE.

The challenge is in defining the expression for the log density; [R] **mlexp** provides many examples. For the Poisson model, the log density is  $f(y_i|\mathbf{x}_i) = -\exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!$ . An explicit definition for our example is the command

```

. * ML estimation (command mlexp) for Poisson model
. mlexp (-exp({xb: private chronic female income _cons}) + docvis*{xb:} -
>          lnfactorial(docvis)), vce(robust)

initial:      log pseudolikelihood = -33899.609
alternative:   log pseudolikelihood = -28031.767
rescale:       log pseudolikelihood = -24020.669
Iteration 0:   log pseudolikelihood = -24020.669
Iteration 1:   log pseudolikelihood = -23995.423
Iteration 2:   log pseudolikelihood = -18539.168
Iteration 3:   log pseudolikelihood = -18503.596
Iteration 4:   log pseudolikelihood = -18503.549
Iteration 5:   log pseudolikelihood = -18503.549

Maximum likelihood estimation
Log pseudolikelihood = -18503.549                                         Number of obs = 4,412

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
xb					
private	.7986654	.1090015	7.33	0.000	.5850265 1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167 1.201614
female	.4925481	.0585365	8.41	0.000	.3778187 .6072775
income	.003557	.0010825	3.29	0.001	.0014354 .0056787
_cons	-.2297264	.1108733	-2.07	0.038	-.4470339 -.0124188

The results are the same as those for the `poisson` command.

Compared with the more general `ml` command and with the functions `optimize()` and `moptimize()`, the `mlexp` command has the advantages of supporting most postestimation commands such as the `margins` command.

### 13.3.4 Postestimation commands

The `ereturn list` command details the estimation results that are stored in `e()`; see section [1.6.2](#). These include regression coefficients in `e(b)` and the estimated VCE in `e(V)`.

Standard postestimation commands available after most estimators are `predict`, `predictnl`, and `margins` for prediction and MES (this chapter); `test`, `testnl`, `lincom`, and `nlcom` for Wald tests and confidence intervals; `linktest` for a model-specification test (section [11.3](#)); and `estimates` for storing results (section [3.5.6](#)).

The `estat vce` command displays the estimate of the VCE, and the `correlation` option displays the correlations for this matrix. The `estat summarize` command summarizes the current estimation sample. The `estat ic` command obtains information criteria (section 13.8.2). More task-specific commands, usually beginning with `estat`, are available for model-specification testing.

Table 3.1 summarizes basic postestimation commands for `regress` that are generally also available after nonlinear model estimation commands. To find the specific postestimation commands available after a command, for example, `poisson`, see [R] **poisson postestimation** or type `help poisson postestimation`.

### 13.3.5 Nonlinear least squares

NLS estimators minimize the sum of squared residuals, so for independent observations, the NLS estimator  $\hat{\beta}$  minimizes

$$Q(\beta) = \sum_{i=1}^N \{y_i - m(\mathbf{x}_i, \beta)\}^2$$

where  $m(\mathbf{x}, \beta)$  is the specified functional form for  $E(y|\mathbf{x})$ , the conditional mean of  $y$  given  $\mathbf{x}$ .

If the conditional mean function is correctly specified, then the NLS estimator is consistent and asymptotically normally distributed. If the data-generating process (DGP) is  $y_i = m(\mathbf{x}_i, \beta) + u_i$ , where  $u_i \sim N(0, \sigma^2)$ , then NLS is fully efficient. If  $u_i \sim [0, \sigma^2]$ , then the NLS default estimate of the VCE is correct; otherwise, a robust estimate should be used.

### 13.3.6 The `nl` command

The `nl` command implements NLS regression. The simplest form of the command directly defines the conditional mean rather than calling a program or function. The syntax is

```
nl (depvar=<sexp>) [if] [in] [weight] [, options]
```

where  $<\text{sexp}>$  is a substitutable expression for the conditional mean. The only relevant option for our analysis here is the `vce()` option for the type of estimate of the VCE.

There are several ways to define the expression for the conditional mean  $\exp(\mathbf{x}'\boldsymbol{\beta})$ ; see [R] **nl**. We present two ways to do so.

The first example of `nl` uses a lengthier expression that gives the parameter names in braces, `{ }`. Additionally, to obtain an analysis-of-variance table, we suppress the usual use of the `vce(robust)` option.

```
. * Nonlinear least-squares regression (command nl) with default standard errors
. nl (docvis = exp({private}*private + {chronic}*chronic
>      + {female}*female + {income}*income + {intercept}))
```

Iteration 0: residual SS = 251743.9  
 Iteration 1: residual SS = 242727.6  
 Iteration 2: residual SS = 241818.1  
 Iteration 3: residual SS = 241815.4  
 Iteration 4: residual SS = 241815.4  
 Iteration 5: residual SS = 241815.4  
 Iteration 6: residual SS = 241815.4  
 Iteration 7: residual SS = 241815.4  
 Iteration 8: residual SS = 241815.4

Source	SS	df	MS	Number of obs	=	4,412
Model	105898.64	5	21179.7289	R-squared	=	0.3046
Residual	241815.36	4407	54.870741	Adj R-squared	=	0.3038
Total	347714	4412	78.8109701	Root MSE	=	7.407479
				Res. dev.	=	30185.68
docvis	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
/private	.7105104	.1170408	6.07	0.000	.4810517	.939969
/chronic	1.057318	.0610386	17.32	0.000	.9376517	1.176984
/female	.4320225	.0523199	8.26	0.000	.3294491	.5345958
/income	.002558	.0006941	3.69	0.000	.0011972	.0039189
/intercept	-.040563	.1272218	-0.32	0.750	-.2899816	.2088557

The `nl` coefficient estimates are similar to those from `poisson` (within 15% for all regressors except `income`). The model diagnostic statistics given include  $R^2$  computed as the model (or explained) sum of squares divided by the total sum of squares, the root mean squared error (MSE) that is the

estimate  $s$  of the standard deviation  $\sigma$  of the model error, and the residual deviance defined in section [13.8.3](#) that is a goodness-of-fit measure used mostly in the GLM literature.

We next give a shorter equivalent expression for the conditional mean function. Also, the `vce(robust)` option is used to allow for heteroskedastic errors, and the `nolog` option is used to suppress the iteration log. We have

```
. * Nonlinear least-squares regression - alternative form of command nl
. nl (docvis = exp({xb: private chronic female income} + {intercept})),
>      vce(robust) nolog

Nonlinear regression                                         Number of obs =      4,412
                                                               R-squared =       0.3046
                                                               Adj R-squared =   0.3038
                                                               Root MSE =      7.407479
                                                               Res. dev. =    30185.68
```

docvis	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
/xb_private	.7105104	.1086194	6.54	0.000	.4975618	.923459
/xb_chronic	1.057318	.0558352	18.94	0.000	.947853	1.166783
/xb_female	.4320225	.0694662	6.22	0.000	.2958338	.5682112
/xb_income	.002558	.0012544	2.04	0.041	.0000988	.0050173
/intercept	-.040563	.1126215	-0.36	0.719	-.2613578	.1802319

The output is the same except for the standard errors, which are now robust to heteroskedasticity. Compared with the `poisson` command robust standard errors, the `nl` robust standard errors are 19% higher for `female` and 16% higher for `income` and are similar for the remaining regressors.

### 13.3.7 Generalized linear models

The GLM framework is the standard nonlinear model framework in many areas of applied statistics, most notably, biostatistics.

The GLM is little used in econometrics. However, for structural parametric nonlinear models such as nonlinear models with selection or nonlinear multilevel models, the Stata command `gsem`, detailed in section 23.6, is restricted to GLMs, in which case some familiarity with GLMs is useful.

GLM estimators are a subset of ML estimators that are based on a density in the LEF, introduced in section [13.3.1](#). They are essentially generalizations of NLS, optimal for a nonlinear regression model with homoskedastic additive errors, to other types of data where there is not only intrinsic heteroskedasticity but also a natural starting point for modeling the intrinsic heteroskedasticity. For example, for the Poisson, the variance equals the mean, and for a binary variable, the variance equals the mean times unity minus the mean.

A quite general GLM estimator  $\hat{\boldsymbol{\theta}}$  maximizes the LEF log likelihood

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^N [a\{m(\mathbf{x}_i, \boldsymbol{\beta})\} + b(y_i) + c\{m(\mathbf{x}_i, \boldsymbol{\beta})\}y_i]$$

where  $m(\mathbf{x}, \boldsymbol{\beta}) = E(y|\mathbf{x})$  is the conditional mean of  $y$ , different specified forms of the functions  $a(\cdot)$  and  $c(\cdot)$  correspond to different members of the LEF, and  $b(\cdot)$  is a normalizing constant. For the Poisson,  $a(\mu) = -\mu$  and  $c(\mu) = \ln \mu$ .

Given definitions of  $a(\mu)$  and  $c(\mu)$ , the mean and variance are necessarily  $E(y) = \mu = -a'(\mu)/c'(\mu)$  and  $\text{Var}(y) = 1/c'(\mu)$ . For the Poisson,  $a'(\mu) = -1$  and  $c'(\mu) = 1/\mu$ , so  $E(y) = 1/(1/\mu) = \mu$  and  $\text{Var}(y) = 1/c'(\mu) = 1/(1/\mu) = \mu$ . This is the variance–mean equality property of the Poisson.

GLM estimators have the important property that in basic cross-sectional applications they are consistent provided only that the conditional mean function is correctly specified. This result arises because the first-order conditions  $\partial Q(\boldsymbol{\theta})/\partial \boldsymbol{\theta} = \mathbf{0}$  can be written as

$$\frac{1}{N} \sum_{i=1}^N c'(\mu_i)(y_i - \mu_i) \frac{\partial \mu_i}{\partial \boldsymbol{\beta}} = \mathbf{0}$$

where  $\mu_i = m(\mathbf{x}_i, \boldsymbol{\beta})$ . It follows that estimator consistency requires only that  $E(y_i - \mu_i) = 0$  or that  $E(y_i|\mathbf{x}_i) = m(\mathbf{x}_i, \boldsymbol{\beta})$ . However, unless the variance is correctly specified [that is,  $\text{Var}(y) = 1/c'(\mu)$ ], we should obtain a robust estimate of the VCE.

### 13.3.8 The `glm` command

The GLM estimator can be computed by using the `glm` command. This command is restricted to a conditional mean function that is of single-index form, and, for historical reasons, models are defined in terms of the inverse of the conditional mean function, called the link function. Thus, a GLM specifies that  $y_i$  has conditional mean

$$E(y_i|\mathbf{x}_i) = g^{-1}(\mathbf{x}'_i \boldsymbol{\beta})$$

where  $g(\cdot)$  is called the link function. For example, for Poisson regression,  $E(y_i|\mathbf{x}_i) = \exp(\mathbf{x}'_{ij} \boldsymbol{\beta})$ , and the link function  $g(\cdot)$  is  $\ln(\cdot)$ , the inverse of the exponential function.

The `glm` command has the syntax

```
glm depvar [indepvars] [if] [in] [weight] [, options]
```

The key options are `family()` to define the particular member of the LEF to be considered and `link()`, where the link function is the inverse of the conditional mean function. The `family()` options are `gaussian(normal)`, `igaussian(inverse Gaussian)`, `binomial(Bernoulli and binomial)`, `poisson(Poisson)`, `nbinomial(negative binomial)`, and `gamma(gamma)`. Different families permit different link functions.

The Poisson estimator can be obtained by using the options `family(poisson)` and `link(log)`. The link function is the natural logarithm because this is the inverse of the exponential function for the conditional mean. We again use the `vce(robust)` option. We expect the same results as those from `poisson` with the `vce(robust)` option.

```

. * Generalized linear models regression for poisson (command glm)
. glm docvis private chronic female income,
>     family(poisson) link(log) vce(robust) nolog
Generalized linear models                                         Number of obs = 4,412
Optimization : ML                                              Residual df = 4,407
                                                               Scale parameter = 1
Deviance          = 28131.11439                               (1/df) Deviance = 6.38328
Pearson           = 57126.23793                               (1/df) Pearson = 12.96261
Variance function: V(u) = u                                     [Poisson]
Link function     : g(u) = ln(u)                                [Log]
                                                               AIC = 8.390095
Log pseudolikelihood = -18503.54883                          BIC = -8852.797

```

docvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	.7986653	.1090014	7.33	0.000	.5850264	1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167	1.201614
female	.4925481	.0585365	8.41	0.000	.3778187	.6072774
income	.003557	.0010825	3.29	0.001	.0014354	.0056787
_cons	-.2297263	.1108733	-2.07	0.038	-.4470339	-.0124187

The results are exactly the same as those given in section [13.3.2](#) for the Poisson quasi-MLE, aside from additional diagnostic statistics (deviance, Pearson) defined in section [13.8.3](#) that are used in the GLM literature. Robust standard errors are used because they do not impose the Poisson density restriction of variance–mean equality.

A standard statistics reference is [McCullagh and Nelder \(1989\)](#); [Hardin and Hilbe \(2018\)](#) present Stata for GLM, and an econometrics reference that covers GLM in some detail is [Cameron and Trivedi \(2013\)](#).

### 13.3.9 Generalized method of moments

GMM estimators minimize an objective function that is more complicated than [\(13.3\)](#) because it is a quadratic form in sums.

The GMM begins with the population moment conditions

$$E\{\mathbf{h}(\mathbf{w}_i, \boldsymbol{\theta})\} = \mathbf{0} \tag{13.4}$$

where  $\theta$  is a  $q \times 1$  vector,  $\mathbf{h}(\cdot)$  is an  $r \times 1$  vector function with  $r \geq q$ , and the vector  $\mathbf{w}_i$  represents all observables, including the dependent variable, regressors, and, where relevant, instrumental variables (IV). A leading example is linear IV (see section 7.3), where  $\mathbf{h}(\mathbf{w}_i, \theta) = \mathbf{z}_i(y_i - \mathbf{x}'_i \beta)$ .

If  $r = q$ , then the method-of-moments (MM) estimator  $\hat{\theta}_{\text{MM}}$  solves the corresponding sample moment condition  $N^{-1} \sum_i \mathbf{h}(\mathbf{w}_i, \theta) = \mathbf{0}$ . This is not possible if  $r > q$ , such as for an overidentified linear IV model, because there are then more equations than parameters.

The GMM estimator  $\hat{\theta}_{\text{GMM}}$  minimizes a quadratic form in  $\sum_i \mathbf{h}(\mathbf{w}_i, \theta)$ , with the objective function

$$Q(\theta) = \left\{ \sum_{i=1}^N \mathbf{h}(\mathbf{w}_i, \theta) \right\}' \mathbf{W} \left\{ \sum_{i=1}^N \mathbf{h}(\mathbf{w}_i, \theta) \right\} \quad (13.5)$$

where the  $r \times r$  weighting matrix  $\mathbf{W}$  is positive-definite symmetric and possibly stochastic with a finite probability limit and does not depend on  $\theta$ . The MM estimator, the special case  $r = q$ , can be obtained most simply by letting  $\mathbf{W} = \mathbf{I}$ , or any other value, and then  $Q(\theta) = 0$  at the optimum.

Provided that condition (13.4) holds, the GMM estimator is consistent for  $\theta$  and is asymptotically normal with the robust estimate of the VCE

$$\hat{V}(\hat{\theta}_{\text{GMM}}) = (\hat{\mathbf{G}}' \mathbf{W} \hat{\mathbf{G}})^{-1} \hat{\mathbf{G}}' \mathbf{W} \hat{\mathbf{S}} \mathbf{W} \hat{\mathbf{G}} (\hat{\mathbf{G}}' \mathbf{W} \hat{\mathbf{G}})^{-1}$$

where, assuming independence over  $i$ ,

$$\widehat{\mathbf{G}} = \sum_{i=1}^N \frac{\partial \mathbf{h}_i}{\partial \boldsymbol{\theta}'} \Big|_{\widehat{\boldsymbol{\theta}}} \quad (13.6)$$

$$\widehat{\mathbf{S}} = \sum_{i=1}^N \mathbf{h}_i (\widehat{\boldsymbol{\theta}}) \mathbf{h}_i (\widehat{\boldsymbol{\theta}})'$$

For MM, the variance simplifies to  $(\widehat{\mathbf{G}}' \widehat{\mathbf{S}}^{-1} \widehat{\mathbf{G}})^{-1}$  regardless of the choice of  $\mathbf{W}$ . For GMM, different choices of  $\mathbf{W}$  lead to different estimators. The best choice of  $\mathbf{W}$  is  $\mathbf{W} = \widehat{\mathbf{S}}^{-1}$ , in which case again the variance simplifies to  $(\widehat{\mathbf{G}}' \widehat{\mathbf{S}}^{-1} \widehat{\mathbf{G}})^{-1}$ . For linear IV, an explicit formula for the estimator can be obtained; see section [7.3](#).

### 13.3.10 The gmm command

GMM estimators minimize an objective function that is a quadratic form in sums; see [\(13.5\)](#). Optimization is more complicated than the single sum for  $m$ -estimators given in [\(13.3\)](#). For some linear models, there are official Stata commands, notably, `ivregress gmm` for cross-sectional data and `xtabond` for dynamic panel data. There are no official commands for specific nonlinear models. Instead, we use the `gmm` command.

The simplest form of the `gmm` command directly defines the conditional mean and has the syntax

```
gmm ([eqname1:]<mexp_1>) ([eqname2:]<mexp_2>) ... [if] [in] [weight]
    [, options]
```

where  $<mexp_j>$  is a substitutable expression for the  $j$ th moment equation. Options include `instruments()` to define the instruments; one of the `onestep`, `twostep` (the default), and `igmm` options for, respectively, one-step, two-step, and iterated GMM estimation; and `wmatrix()` to define the weighting matrix if estimation is by two-step or iterated GMM for an overidentified model. For a just-identified model, these different estimation methods lead to the same estimates. For models that are more complicated to specify, we use a variant of the `gmm` command that references a separate user-written program that defines the moment conditions.

Postestimation commands following `gmm` include `estat overid` for an overidentified model and the `margins` command; see the example below for an example of computation of MES.

We apply the `gmm` command to the nonlinear IV estimator for the Poisson model with an endogenous regressor. The Poisson regression model specifies that  $E\{y - \exp(\mathbf{x}'\boldsymbol{\beta})|\mathbf{x}\} = 0$  because  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ . Suppose instead that  $E\{y - \exp(\mathbf{x}'\boldsymbol{\beta})|\mathbf{x}\} \neq 0$ , because of endogeneity of one or more regressors, but that there are instruments  $\mathbf{z}$  such that

$$E[\mathbf{z}_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\}] = \mathbf{0} \quad (13.7)$$

This defines the moment condition in (13.4), and the GMM estimator then minimizes the quadratic form

$$Q(\boldsymbol{\beta}) = \left[ \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \right]' \mathbf{W} \left[ \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \right]$$

where different estimation methods and choices of the weighting matrix  $\mathbf{W}$  lead to different estimates if the model is overidentified (here  $\mathbf{z}_i$  has more entries than  $\mathbf{x}_i$ ).

We use the dependent and independent variables to define a substitutable expression  $\{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\}$ , using a syntax that is similar to that for the `n1` command. We use the `instruments()` option to define the variables to be used in the instruments  $\mathbf{z}_i$ . We continue with the number of doctor-visits regression example, except that we now treat the regressor `private` as endogenous, with single instrument `firmsize` (measured in hundreds of employees). We have

```

. * Command gmm for GMM estimation (nonlinear IV) for Poisson model
. gmm (docvis - exp({xb:private chronic female income _cons})),
> instruments(firmsize chronic female income) onestep nolog
Final GMM criterion Q(b) = 1.29e-17
note: model is exactly identified.
GMM estimation
Number of parameters = 5
Number of moments = 5
Initial weight matrix: Unadjusted Number of obs = 4,412

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
private	1.340292	1.559015	0.86	0.390	-1.715322 4.395905
chronic	1.072908	.0762684	14.07	0.000	.9234242 1.222391
female	.4778178	.0690393	6.92	0.000	.3425032 .6131323
income	.0027833	.002192	1.27	0.204	-.0015129 .0070795
_cons	-.6832462	1.349606	-0.51	0.613	-3.328424 1.961932

Instruments for equation 1: firmsize chronic female income \_cons

By default, robust standard errors are computed. The biggest change compared with the Poisson regression output given in section [13.3.2](#) is for the endogenous regressor `private`. This regressor is now much less precisely estimated, with a standard error of 1.559 compared with 0.109 and a coefficient that has increased substantially from 0.798 to 1.340, though it is now statistically insignificant. Similar efficiency loss in estimation of endogenous regressors often occurs with linear IV estimation using cross-sectional data; see the example in section [7.4.6](#).

The following command, explained in section [13.7.8](#), computes the average marginal effects (AMEs) of changes in the expected number of doctor visits as regressors change.

```

. * Computation of AMEs following for this gmm example
. margins, dydx(*) expression(exp(predict(xb)))
(output omitted)

```

The preceding example was a just-identified model with the same number of instruments as regressors. Thus, the minimized value of the objective function is zero (here  $1.29 \times 10^{-17}$ , reflecting numerical rounding error). For application of the `gmm` command to an overidentified model, see section 20.7.3. Also, [R] `gmm` provides many more examples.

### 13.3.11 The gmm command for two-step estimators

The preceding example entailed estimation based on the single moment orthogonality condition (13.7). The `gmm` command can also be applied to multiple moment conditions.

As an example, we consider the control function estimator for linear regression with a single endogenous regressor (see section 7.4.7), which is an example of a two-step sequential estimator. The structural model is  $y = \mathbf{x}'\beta + u$ , where  $\mathbf{x}$  includes the endogenous variable  $y_2$ . The first-stage model is  $y_2 = \mathbf{z}'\pi + v$ , where  $\mathbf{z}$  includes the exogenous regressors in the structural model  $y$  and additional instruments necessary for identification.

The control function estimator first ordinary least-squares (OLS) regresses  $y_2$  on  $\mathbf{z}$ , yielding residual  $\hat{v} = y_2 - \mathbf{z}'\hat{\pi}$ , and then OLS regresses  $y$  on  $\mathbf{x}$  and  $\hat{v}$ , yielding estimates that can be shown to be equal to the two-stage least-squares (2SLS) estimates. The corresponding sample moment conditions are

$$\begin{aligned} \sum_i \mathbf{z}_i(y_{2i} - \mathbf{z}'_i\pi) &= \mathbf{0} \\ \sum_i \begin{bmatrix} \mathbf{x}_i \\ y_{2i} - \mathbf{z}'_i\pi \end{bmatrix} \{y_i - \mathbf{x}'_i\beta - \gamma(y_{2i} - \mathbf{z}'_i\pi)\} &= \mathbf{0} \end{aligned} \quad (13.8)$$

The `gmm` command requires that the instruments provided in the `instruments()` option depend only on observed variables, and not unknown parameters. To enable this, we reexpress (13.8) as

$$\begin{aligned} \sum_i \mathbf{z}_i(y_{2i} - \mathbf{z}'_i\pi) &= \mathbf{0} \\ \sum_i \mathbf{x}_i \{y_i - \mathbf{x}'_i\beta - \gamma(y_{2i} - \mathbf{z}'_i\pi)\} &= \mathbf{0} \\ \sum_i 1 \times [(y_{2i} - \mathbf{z}'_i\pi) \{y_i - \mathbf{x}'_i\beta - \gamma(y_{2i} - \mathbf{z}'_i\pi)\}] &= 0 \end{aligned}$$

so the instruments are, respectively,  $\mathbf{z}$ ,  $\mathbf{x}$ , and 1.

We apply this method to the example of section [7.4.7](#). We first define globals that make the code easier to read.

```
. * Data and globals for gmm estimation of chapter 7 control function example
. qui use mus207mepspresdrugs, clear
. global y1 ldrugexp          // Dependent variable
. global y2 hi_empunion      // Endogenous regressor
. global x2list totchr age female blhisp linc // Exogenous regressors
. global xlist ssiratio $x2list // Structural equation regressors
. global zlist $y2 $x2list    // First-stage regressors
```

We then obtain the estimates

```

. * Command gmm for estimation of control function linear model
. gmm (eq1: ($y2 - {zpi:ssiratio $x2list _cons} ) )
>   (eq2: ($y1 - {xb:$zlist _cons} -
>     {gamma}*( $y2 - {zpi:})) )
>   (eq3: (( $y2 -{zpi:}) * ($y1 - {xb:}
>     -{gamma}*( $y2 - {zpi:})) ), 
>   instruments(eq1: $xlist) instruments(eq2: $zlist)
>   instruments(eq3: ) onestep winitial(unadjusted, independent) nolog
Final GMM criterion Q(b) =  2.96e-32
note: model is exactly identified.

GMM estimation

Number of parameters =  15
Number of moments    =  15
Initial weight matrix: Unadjusted                               Number of obs      = 10,068

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
zpi	ssiratio	-.2205333	.0151123	-14.59	0.000 -.2501529 -.1909137
	totchr	.0133494	.0036373	3.67	0.000 .0062205 .0204783
	age	-.0084926	.0007038	-12.07	0.000 -.0098721 -.0071131
	female	-.0721048	.0096345	-7.48	0.000 -.0909881 -.0532215
	blhisp	-.0609939	.012235	-4.99	0.000 -.0849741 -.0370137
	linc	.0454804	.0061999	7.34	0.000 .0333288 .057632
	_cons	1.039144	.0574802	18.08	0.000 .9264845 1.151803
xb	hi_empunion	-.8115015	.1884024	-4.31	0.000 -1.180763 -.4422396
	totchr	.4492135	.0100423	44.73	0.000 .429531 .468896
	age	-.0122415	.0027711	-4.42	0.000 -.0176727 -.0068102
	female	-.0140507	.0310339	-0.45	0.651 -.0748761 .0467747
	blhisp	-.2089573	.0383529	-5.45	0.000 -.2841276 -.1337869
	linc	.0815472	.0207576	3.93	0.000 .0408631 .1222314
	_cons	6.692548	.2449266	27.32	0.000 6.212501 7.172595
/gamma	.9039114	.1900969	4.76	0.000 .5313283 1.276495	

Instruments for equation eq1: ssiratio totchr age female blhisp linc \_cons  
Instruments for equation eq2: hi\_empunion totchr age female blhisp linc \_cons  
Instruments for equation eq3: \_cons

The estimates are identical to those in section [7.4.7](#), except the correct heteroskedastic-robust standard errors that control for the first-stage estimation are computed. For the linear model, the control function estimator equals the 2SLS estimator, and, as expected, the estimates and standard errors are identical to those for 2SLS given in section [7.4.4](#).

This example extends to control function estimators in nonlinear models with an endogenous regressor. More generally, the method can be applied to two-step estimators, such as Heckman's two-step estimator for the sample-selection model in section 19.6.4. [Drukker \(2014\)](#) outlines this method using the `gmm` command for a treatment-effects example. The method is used for many of Stata's treatment-effects estimators; see section 24.6.12.

### 13.3.12 Other estimators

The preceding part of this chapter covers most of the estimators used in microeconomics analysis using cross-sectional data. We now consider some nonlinear estimators that are not covered.

One approach is to specify a linear function for  $E\{h(y)|\mathbf{x}\}$ , so  $E\{h(y)|\mathbf{x}\} = \mathbf{x}'\boldsymbol{\beta}$ , where  $h(\cdot)$  is a nonlinear function. An example is the Box–Cox transformation in section [3.6](#). A disadvantage of this alternative approach is the transformation bias that arises if we then wish to predict  $y$  or  $E(y|\mathbf{x})$ .

Nonparametric and semiparametric estimators do not completely specify the functional forms of key model components such as  $E(y|\mathbf{x})$ . Several methods for nonparametric regression of  $y$  on a scalar  $x$ , including the `lpoly`, `npregress`, and `lowess` commands, are introduced in sections [2.6.6](#) and [14.6](#), and chapter 27 provides a more thorough treatment. Nonlinear methods for clustered data are summarized in section [13.9](#), and nonlinear methods for panel data are presented in chapter 22.

## 13.4 Different estimates of the VCE

Given an estimator, there are several different standard methods for computation of standard errors and subsequent test statistics and confidence intervals. The most commonly used methods yield default, robust, and cluster-robust standard errors. This section extends the results in section [3.4](#) for the OLS estimator to nonlinear estimators.

### 13.4.1 General framework

We consider inference for the estimator  $\hat{\theta}$  of a  $q \times 1$  parameter vector  $\theta$  that solves the  $q$  equations

$$\sum_{i=1}^N \mathbf{g}_i(\hat{\theta}) = \mathbf{0} \quad (13.9)$$

where  $\mathbf{g}_i(\cdot)$  is a  $q \times 1$  vector. For  $m$  estimators defined in section [13.3](#), differentiation of objective function [\(13.3\)](#) leads to first-order conditions with  $\mathbf{g}_i(\theta) = \partial q_i(y_i, \mathbf{x}_i, \theta) / \partial \theta$ . It is assumed that

$$E\{\mathbf{g}_i(\theta)\} = \mathbf{0}$$

a condition that for standard estimators is necessary and sufficient for consistency of  $\hat{\theta}$ .

This setup covers most models and estimators, with the notable exception of the overidentified 2SLS and GMM estimators presented in chapter [7](#) in the linear case, and section [13.3.9](#) in the nonlinear case.

Under appropriate assumptions, it can be shown that

$$\widehat{\boldsymbol{\theta}} \stackrel{a}{\sim} N \left\{ \boldsymbol{\theta}, \text{Var} \left( \widehat{\boldsymbol{\theta}} \right) \right\}$$

where  $\text{Var}(\widehat{\boldsymbol{\theta}})$  denotes the (asymptotic) VCE. Furthermore,

$$\begin{aligned} \text{Var} \left( \widehat{\boldsymbol{\theta}} \right) &= \\ \left[ E \left\{ \sum_i \mathbf{H}_i(\boldsymbol{\theta}) \right\} \right]^{-1} E \left\{ \sum_i \sum_j \mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_j(\boldsymbol{\theta})' \right\} \left[ E \left\{ \sum_i \mathbf{H}_i(\boldsymbol{\theta})' \right\} \right]^{-1} \end{aligned} \quad (13.10)$$

where  $\mathbf{H}_i(\boldsymbol{\theta}) = \partial \mathbf{g}_i / \partial \boldsymbol{\theta}'$ . This general expression for  $\text{Var}(\widehat{\boldsymbol{\theta}})$  is said to be of “sandwich form” because it can be written as  $\mathbf{A}^{-1} \mathbf{B} \mathbf{A}'^{-1}$ , with  $\mathbf{B}$  sandwiched between  $\mathbf{A}^{-1}$  and  $\mathbf{A}'^{-1}$ . OLS is a special case with  $\mathbf{g}_i(\widehat{\boldsymbol{\beta}}) = \mathbf{x}_i \widehat{u}_i = \mathbf{x}_i(y_i - \mathbf{x}'_i \widehat{\boldsymbol{\beta}})$  and  $\mathbf{H}_i(\boldsymbol{\beta}) = \mathbf{x}_i \mathbf{x}'_i$ .

We wish to obtain the estimated asymptotic variance matrix  $\widehat{V}(\widehat{\boldsymbol{\theta}})$  and the associated standard errors, which are the square roots of the diagonal entries of  $\widehat{V}(\widehat{\boldsymbol{\theta}})$ . This obviously entails replacing  $\boldsymbol{\theta}$  with  $\widehat{\boldsymbol{\theta}}$ . The first and third matrices in (13.10) can be estimated using  $\widehat{\mathbf{A}} = \sum_i \mathbf{H}_i(\widehat{\boldsymbol{\theta}})$ . But estimation of  $E \left\{ \sum_i \sum_j \mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_j(\boldsymbol{\theta})' \right\}$  requires additional distributional assumptions, such as independence over  $i$  and possibly a functional form for  $E \{ \mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_i(\boldsymbol{\theta})' \}$ . [Note that the obvious  $\sum_i \sum_j \mathbf{g}_i(\widehat{\boldsymbol{\theta}}) \mathbf{g}_j(\widehat{\boldsymbol{\theta}})' = \mathbf{0}$  because from (13.9)  $\sum_i \mathbf{g}_i(\widehat{\boldsymbol{\theta}}) = \mathbf{0}$ .]

### 13.4.2 The vce() option

Different assumptions lead to different estimates of the VCE. They are obtained by using the `vce(vcetype)` option for the estimation command being used. The specific *vcetypes* available vary with the estimation command. Their formulas are detailed in subsequent sections.

For the `poisson` command, many *vcetypes* are supported.

The `vce(oim)` and `vce(opg)` options use the DGP assumptions to evaluate the expectations in (13.10); see section 13.4.4. The `vce(oim)` option is the

default.

The `vce(robust)` and `vce(cluster clustvar)` options use sandwich estimators that do not use the DGP assumptions to explicitly evaluate the expectations in (13.10). The `vce(robust)` option assumes independence over  $i$ . The `vce(cluster clustvar)` option permits a limited form of correlation over  $i$ , within clusters, where the clusters are independent and there are many clusters; see section 13.4.6. For commands that already control for clustering, such as `xtreg`, the `vce(robust)` option provides a cluster-robust estimate of the VCE.

The `vce(bootstrap)` and `vce(jackknife)` options use resampling schemes that make limited assumptions on the DGP similar to those for the option `vce(robust)` or `vce(cluster clustvar)`; see section 13.4.8.

The various `vce()` options need to be used with considerable caution. Estimates of the VCE other than the default estimate are used when some part of the DGP is felt to be misspecified. Because this is likely to be the case, it is standard to use default standard errors that are robust, usually heteroskedastic robust for independent observations and cluster-robust for clustered observations. At the same time, in many instances, misspecification of the DGP leads to inconsistent parameter estimates; see section 13.3.1.

### 13.4.3 Application of the `vce()` option

For count data, the natural starting point is the MLE, assuming a Poisson distribution. It can be shown that the default ML standard errors are based on the Poisson distribution restriction of variance-mean inequality. But in practice, count data are often “overdispersed” with  $\text{Var}(y|\mathbf{x}) > \exp(\mathbf{x}'\boldsymbol{\beta})$ , in which case the default ML standard errors can be shown to be biased downward. At the same time, the Poisson MLE can be shown to be consistent provided only that  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$  is the correct specification of the conditional mean.

These considerations make the Poisson MLE a leading candidate for using the option `vce(robust)` rather than the default. The `vce(cluster clustvar)` option assumes independence over clusters, however clusters are

defined, rather than independence over  $i$ . The `vce(bootstrap)` estimate is asymptotically equivalent to the `vce(robust)` estimate.

For the Poisson MLE with  $K$  parameters, it can be shown that the default, robust, and cluster-robust estimates of the VCE are given by, respectively,

$$\begin{aligned}\widehat{V}_{\text{oim}}(\widehat{\beta}) &= \left( \sum_i e^{\mathbf{x}'_i \widehat{\beta}} \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \\ \widehat{V}_{\text{rob}}(\widehat{\beta}) &= \left( \sum_i e^{\mathbf{x}'_i \widehat{\beta}} \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \left\{ \frac{N}{N-K} \sum_i (y_i - e^{\mathbf{x}'_i \widehat{\beta}})^2 \mathbf{x}_i \mathbf{x}'_i \right\} \left( \sum_i e^{\mathbf{x}'_i \widehat{\beta}} \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \\ \widehat{V}_{\text{clu}}(\widehat{\beta}) &= \left( \sum_i e^{\mathbf{x}'_i \widehat{\beta}} \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \left( \frac{C}{C-1} \sum_c \widehat{\mathbf{g}}_c \widehat{\mathbf{g}}'_c \right) \left( \sum_i e^{\mathbf{x}'_i \widehat{\beta}} \mathbf{x}_i \mathbf{x}'_i \right)^{-1}\end{aligned}$$

where  $\widehat{\mathbf{g}}_c = \sum_{i:i \in c} (y_i - e^{\mathbf{x}'_i \widehat{\beta}}) \mathbf{x}_i$ ,  $c = 1, \dots, C$  denotes the clusters and  $N/(N-K)$  and  $C/(C-1)$  are degrees-of-freedom adjustments used by Stata.

Implementation is straightforward, except that in this example, there is no natural reason for clustering. For illustrative purposes, we cluster on `age`, in which case we are assuming correlation across individuals of the same age and independence of individuals of different age. For the bootstrap, we first set the seed, for replicability, and set the number of replications at 400, considerably higher than the Stata default. We obtain

```

. * Different VCE estimates after Poisson regression
. qui use mus210mepsdocvisyoung, clear
. keep if year02==1
(25,712 observations deleted)
. qui poisson docvis private chronic female income
. estimates store VCE_oim
. qui poisson docvis private chronic female income, vce(opg)
. estimates store VCE_opg
. qui poisson docvis private chronic female income, vce(robust)
. estimates store VCE_rob
. qui poisson docvis private chronic female income, vce(cluster age)
. estimates store VCE_clu
. set seed 10101
. qui poisson docvis private chronic female income, vce(bootstrap, reps(400))
. estimates store VCE_boot
. estimates table VCE_oim VCE_opg VCE_rob VCE_clu VCE_boot, b(%8.4f) se

```

Variable	VCE_oim	VCE_opg	VCE_rob	VCE_clu	VCE_boot
private	0.7987 0.0277	0.7987 0.0072	0.7987 0.1090	0.7987 0.1496	0.7987 0.1039
chronic	1.0919 0.0158	1.0919 0.0046	1.0919 0.0560	1.0919 0.0603	1.0919 0.0566
female	0.4925 0.0160	0.4925 0.0046	0.4925 0.0585	0.4925 0.0686	0.4925 0.0594
income	0.0036 0.0002	0.0036 0.0001	0.0036 0.0011	0.0036 0.0012	0.0036 0.0011
_cons	-0.2297 0.0287	-0.2297 0.0075	-0.2297 0.1109	-0.2297 0.1454	-0.2297 0.1044

Legend: b/se

The first two ML-based standard errors, explained in section [13.4.4](#), are very different. This indicates a problem with the assumption of a Poisson density. The third-column robust standard errors are roughly four times the first-column default standard errors. This very large difference often happens when fitting Poisson models. For other estimators, the difference is usually not as great. In particular, for OLS, robust standard errors are often within 20% (higher or lower) of the default. The fourth-column cluster-robust standard errors are 8–37% higher than the robust standard errors. In other applications, the difference can be much larger; see section [3.4.6](#) for a rule of thumb in the linear case. The fifth-column bootstrap standard errors are

within 1% of the third-column robust standard errors, confirming that they are essentially equivalent.

In this example, it would be misleading to use the default standard errors. We should at least use the robust standard errors. This requires relaxing the assumption of a Poisson distribution so that the model should not be used to predict conditional probabilities. But, at least for the Poisson MLE,  $\hat{\beta}$  is a consistent estimate provided that the conditional mean is indeed the specified  $\exp(\mathbf{x}'\beta)$ .

### 13.4.4 Default estimate of the VCE

If no option is used, then we obtain “default” standard errors. These make the strongest assumptions, essentially that all relevant parts of the DGP are specified and are specified correctly. This permits considerable simplification, not given here, that leads to the sandwiched form  $\mathbf{A}^{-1}\mathbf{B}\mathbf{A}'^{-1}$  simplifying to a multiple of  $\mathbf{A}^{-1}$ .

For the MLE (with data independent over  $i$ ), it is assumed that the density is correctly specified. Then the information matrix equality leads to simplification so that

$$\widehat{V}_{\text{def}}(\hat{\theta}) = - \left[ \sum_i E\{\mathbf{H}_i(\theta)\} |_{\hat{\theta}} \right]^{-1}$$

The default `vce(oim)` option, where `oim` is an acronym for the observed information matrix, gives this estimate of the VCE for Stata ML commands. This estimator is also the default for `regress`, in which case it goes under the name `vce(ols)`, yielding  $\widehat{V}_{\text{def}}(\hat{\beta}) = s^2 (\sum_i \mathbf{x}_i \mathbf{x}_i')^{-1}$  with  $s^2 = \sum_i \hat{u}_i^2 / (N - K)$ .

The `vce(opg)` option gives an alternative estimate, called the outer product of the gradient estimate:

$$\widehat{V}_{\text{opg}}(\widehat{\boldsymbol{\theta}}) = \left\{ \sum_i \mathbf{g}_i(\widehat{\boldsymbol{\theta}}) \mathbf{g}_i(\widehat{\boldsymbol{\theta}})' \right\}^{-1}$$

This is asymptotically equivalent to the default estimate if the density is correctly specified.

### 13.4.5 Heteroskedastic-robust estimate of the VCE

The `vce(robust)` option to cross-sectional estimation commands calculates the sandwich estimate under the assumption of independence. Then  $E\{\sum_i \sum_j \mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_j(\boldsymbol{\theta})'\} = E\{\sum_i \mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_i(\boldsymbol{\theta})'\}$ , leading to the vce robust estimate

$$\widehat{V}_{\text{rob}}(\widehat{\boldsymbol{\theta}}) = \left( \sum_i \widehat{\mathbf{H}}_i \right)^{-1} \left( \frac{N}{N-K} \sum_i \widehat{\mathbf{g}}_i \widehat{\mathbf{g}}_i' \right) \left( \sum_i \widehat{\mathbf{H}}_i' \right)^{-1}$$

where  $\widehat{\mathbf{H}}_i = \mathbf{H}_i(\widehat{\boldsymbol{\theta}})$  and  $\widehat{\mathbf{g}}_i = \mathbf{g}_i(\widehat{\boldsymbol{\theta}})$ . In some special cases, such as NLS,  $\widehat{\mathbf{H}}_i$  is replaced by the expected Hessian  $E(\mathbf{H}_i)$  evaluated at  $\widehat{\boldsymbol{\theta}}$ . The factor  $N/(N - K)$  in the middle term is an ad hoc degrees-of-freedom adjustment analogous to that for the linear regression model with independent and identically distributed normal errors. This estimator is a generalization of similar results of [Huber \(1967\)](#) for the MLE and the heteroskedasticity consistent estimate of [White \(1980\)](#) for the OLS estimator. It is often called heteroskedasticity robust rather than robust.

The `vce(robust)` option should be used with caution. It is robust in the sense that, unlike default standard errors, no assumption is made about the functional form for  $E\{\mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_i(\boldsymbol{\theta})'\}$ . But if  $E\{\mathbf{g}_i(\boldsymbol{\theta}) \mathbf{g}_i(\boldsymbol{\theta})'\}$  is misspecified, warranting use of robust standard errors, then it may also be the case that  $E\{\mathbf{g}_i(\boldsymbol{\theta})\} \neq 0$ . Then we have the much more serious problem of  $\widehat{\boldsymbol{\theta}}$  being inconsistent for  $\boldsymbol{\theta}$ . For example, the tobit MLE and the MLE for any other parametric model with selection or truncation become inconsistent as soon as any distributional assumptions are relaxed. The only advantage then of

using the robust estimate of the VCE is that it does give a consistent estimate of the VCE. However, it is the VCE of an inconsistent estimator.

Essentially, the `vce(robust)` option means that the standard errors are robust to model misspecification, provided observations are independent, but does not mean in general that the estimator itself is robust to misspecification.

There are, however, some commonly used estimators that maintain consistency under relatively weak assumptions. ML and GLM estimators based on the LEF (see section [13.3.1](#)) require only that the conditional mean function be correctly specified for parameter estimates to be consistent. And IV estimators are consistent provided only that a valid instrument is used so that the model error  $u_i$  and instrument vector  $\mathbf{z}_i$  satisfy  $E(u_i|\mathbf{z}_i) = \mathbf{0}$ .

The preceding discussion applies to cross-sectional estimation commands. For panel data or clustered data, the `vce(robust)` option for `xt` commands such as `xtreg` produces a cluster-robust estimate of the VCE that we now present.

#### 13.4.6 Cluster-robust estimate of the VCE

A common alternative to independent observations is that observations fall into clusters, where observations in different clusters are independent but observations within the same cluster are no longer independent. For example, individuals may be grouped into villages, with correlation within villages but not across villages. Such regional groupings are especially important to control for if the regressor of interest, such as a policy variable, is invariant within the region. Then, for cross-sectional estimators, the heteroskedastic-robust estimate of the VCE is incorrect and can be substantially downward biased; see section [6.4.2](#) for an empirical example of this consequence of clustered errors.

Instead, we use a cluster-robust estimate of the VCE. The first-order conditions can be summed within cluster and reexpressed as

$$\sum_{c=1}^C \mathbf{g}_c(\hat{\boldsymbol{\theta}}) = \mathbf{0}$$

where  $c$  denotes the  $c$ th cluster, there are  $C$  clusters, and  $\mathbf{g}_c(\boldsymbol{\theta}) = \sum_{i:i \in c} \mathbf{g}_i(\boldsymbol{\theta})$ . The key assumption is that  $E\{\mathbf{g}_i(\boldsymbol{\theta})\mathbf{g}_j(\boldsymbol{\theta})'\} = \mathbf{0}$  if  $i$  and  $j$  are in different clusters. Only minor adaptation of the previous algebra is needed, and we obtain

$$\widehat{V}_{\text{clus}}(\widehat{\boldsymbol{\theta}}) = \left( \sum_c \widehat{\mathbf{H}}_c \right)^{-1} \left( \frac{C}{C-1} \sum_c \widehat{\mathbf{g}}_c \widehat{\mathbf{g}}_c' \right) \left( \sum_c \widehat{\mathbf{H}}_c' \right)^{-1}$$

where  $\mathbf{H}_c(\boldsymbol{\theta}) = \partial \mathbf{g}_c(\boldsymbol{\theta}) / \partial \boldsymbol{\theta}'$ . This estimator was proposed by [Liang and Zeger \(1986\)](#), and the scaling  $C/(C - 1)$  is an ad hoc degrees-of-freedom correction. The estimator assumes that the number of clusters  $C \rightarrow \infty$ . When each cluster has only one observation,  $\widehat{V}_{\text{clus}}(\widehat{\boldsymbol{\theta}}) = \{(N - K)/(N - 1)\}\widehat{V}_{\text{rob}}(\widehat{\boldsymbol{\theta}})$ , the cluster-robust and robust standard errors then differ only by a small degrees-of-freedom correction.

This estimator is obtained by using the `vce(cluster clustvar)` option, where *clustvar* is the name of the variable that defines the cluster, such as a village identifier. For panel data, or clustered data, `xt` commands such as `xtreg` already explicitly allow for clustering in estimation, and the cluster-robust estimate of the VCE is obtained by using the `vce(robust)` option rather than the `vce(cluster clustvar)` option.

The same caveat as in the heteroskedastic-robust case applies. It is still necessary that  $E\{\mathbf{g}_c(\boldsymbol{\theta})\} = \mathbf{0}$  to ensure consistent estimation of  $\boldsymbol{\theta}$ . Essentially, the joint distribution of the  $\mathbf{g}_i(\boldsymbol{\theta})$  within cluster can be misspecified because of assuming independence when there is in fact dependence, but the marginal distribution of  $\mathbf{g}_i(\boldsymbol{\theta})$  must be correctly specified in the sense that  $E\{\mathbf{g}_i(\boldsymbol{\theta})\} = \mathbf{0}$  for each component of the cluster.

An additional caveat is that the validity of  $\widehat{V}_{\text{clus}}(\widehat{\theta})$  is based on asymptotic theory in the number of clusters. When there are few clusters, say, less than 30, the asymptotic theory provides a poor approximation, leading to test overrejection. At a minimum, one should base inference on the  $t(C - 1)$  distribution, rather than use Stata's default for nonlinear commands that uses the standard normal. Even this adjustment leads to overrejection in practice, and this complication is an area of ongoing research. The community-contributed `boottest` command can be used to implement a wild cluster bootstrap following estimation of many models; see section [12.6](#).

### 13.4.7 Heteroskedasticity- and autocorrelation-consistent estimate of the VCE

Heteroskedasticity- and autocorrelation-consistent (HAC) estimates of the VCE, such as the Newey–West ([1987](#)) estimator, are a generalization of the robust estimate to time-series data. This permits some correlation of adjacent observations, up to, say,  $m$  periods apart.

HAC estimates are implemented in Stata for some linear time-series estimators, such as `newey` for OLS and `ivregress` for IV regression. For nonlinear estimators, HAC estimates are available for `n1`, `glm`, and `gmm` by specifying the `vce(hac kernel)` option, in which case you must `tsset` your data.

Spatial HAC estimates that control for error correlation that dampens with distance between observations, rather than difference in time, are presented in section [26.6.1](#).

In microeconomics analysis, panel data have a time-series component. For short panels covering few time periods, there is no need to use HAC estimates. For long panels spanning many time periods, there is more reason to use HAC estimates of the VCE. An example using the community-contributed `xtscc` command is given in section [9.5.5](#).

### 13.4.8 Bootstrap standard errors

Stata estimation commands with the `vce(bootstrap)` option provide standard errors using the bootstrap, specifically, a paired bootstrap. The default bootstrap assumes independent observations and is asymptotically equivalent to computing robust standard errors, provided that the number of bootstraps is large. Similarly, a cluster bootstrap that assumes independence across clusters but not within clusters is equivalent to computing cluster-robust standard errors.

A related option is `vce(jackknife)`. This can be computationally demanding because it involves recomputing the estimator  $N$  times, where  $N$  is the sample size.

These methods were detailed in chapter 12. In that chapter, we also presented different use of the bootstrap to implement a more refined asymptotic theory that can lead to  $t$  statistics with better size properties, and confidence intervals with better coverage, in finite samples. In particular, with few clusters, the wild cluster bootstrap can be implemented using the community-contributed `boottest` command; see section 12.6.

### 13.4.9 Statistical inference

Given a method to estimate the VCE, we can compute standard errors,  $t$  statistics, confidence intervals, and Wald hypothesis tests. These are automatically provided by estimation commands such as `poisson`. Some tests—notably, likelihood-ratio tests—are no longer appropriate once DGP assumptions are relaxed to allow, for example, a robust estimate of the VCE.

More complicated statistical inference can be performed by using the `test`, `testnl`, `lincom`, and `nlcom` commands, which are detailed in section 11.3.

## 13.5 Prediction

Prediction of the conditional mean and forecast of the actual value of  $y$  following linear regression was presented in sections [4.2](#) and [4.3](#).

In this section, we consider prediction following nonlinear estimation. Most often, the prediction is one of the conditional mean  $E(y|\mathbf{x})$ . This can be much more precisely predicted than can the actual value of  $y$  given  $\mathbf{x}$ .

### 13.5.1 The predict and predictnl commands

A new variable that contains the prediction for each observation can be obtained by using the `predict` postestimation command. After single-equation commands, this command has the syntax

```
predict [ type ] newvar [ if ] [ in ] [ , options ]
```

The prediction is stored as the variable *newvar* and is of the data type *type*, the default being single precision. The type of prediction desired is defined with *options*, and several different types of prediction are usually available. The possibilities vary with the preceding estimation command.

After `poisson`, the key option for the `predict` command is the default `n` option. This computes  $\exp(\mathbf{x}'_i \hat{\beta})$ , the predicted expected number of events. The `xb` option calculates the linear prediction  $\mathbf{x}'_i \hat{\beta}$ , and `stdp` calculates  $\{\mathbf{x}'_i \hat{V}(\hat{\beta}) \mathbf{x}_i\}^{1/2}$ , the standard error of  $\mathbf{x}'_i \hat{\beta}$ . The `score` option calculates the derivative of the log likelihood with respect to the linear prediction. For the Poisson MLE, this is  $y_i - \exp(\mathbf{x}'_i \hat{\beta})$  and can be viewed as a Poisson residual. The `pr(a)` option calculates  $\Pr(y = a)$ , and the `pr(a,b)` option calculates  $\Pr(a \leq y \leq b)$ . These last two options are useful only if the Poisson is the correct distribution.

The `predictnl` command enables the user to provide a formula for the prediction. The syntax is

```
predictnl [ type ] newvar = pnl_exp [ if ] [ in ] [ , options ]
```

where *pnl\_exp* is an expression that is illustrated in the next section. The options provide quantities not provided by `predict` that enable Wald statistical inference on the predictions. In particular, the `se(newvar2)` option creates a new variable containing standard errors for the prediction *newvar* for each observation. These standard errors are computed using the delta method detailed in section [11.3.11](#). Other options include `variance()`, `wald()`, `p()`, and `ci()`.

The `predict` and `predictnl` commands act on the currently defined sample, with the `if` and `in` qualifiers used if desired to predict for a subsample. One can inadvertently predict using a sample different from the estimation sample. The `if e(sample)` qualifier ensures that the estimation sample is used in prediction. At other times, it is desired to deliberately use estimates from one sample to predict using a different sample. This can be done by estimating with one sample, reading a new sample into memory, and then predicting using this new sample.

### 13.5.2 Application of `predict` and `predictnl`

The predicted mean number of doctor visits for each individual in the sample can be computed by using the `predict` command with the default option. We use the `if e(sample)` qualifier to ensure that prediction is for the same sample as the estimation sample. This precaution is not necessary here but is good practice to avoid inadvertent error. We also obtain the same prediction by using `predictnl`; the `se()` option additionally obtains the standard error of the prediction. We obtain

```
. * Predicted mean number of doctor visits using predict and predictnl
. qui poisson docvis private chronic female income, vce(robust)
. predict muhat if e(sample), n
. predictnl muhat2 = exp(_b[private]*private + _b[chronic]*chronic
> + _b[female]*female + _b[income]*income + _b[_cons]), se(semuhat2)
. summarize docvis muhat muhat2 semuhat2
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	4,412	3.957389	7.947601	0	134
muhat	4,412	3.957389	2.985057	.7947512	15.48004
muhat2	4,412	3.957389	2.985057	.7947512	15.48004
semuhat2	4,412	.2431483	.1980062	.0881166	3.944615

Here the average of the predictions of  $E(y|\mathbf{x})$  is 3.957, equal to the average of the  $y$  values. This special property holds only for some estimators—OLS, just-identified linear IV, Poisson, logit, and exponential (with exponential conditional mean)—provided that these models include an intercept. The standard deviation of the predictions is 2.985, less than that of  $y$ . The predicted values range from 0.8 to 15.5 compared with a sample range of 0–134.

The model quite precisely estimates  $E(y|\mathbf{x})$  because from the last row, the standard error of  $\exp(\mathbf{x}'_i \hat{\beta})$  as an estimate of  $\exp(\mathbf{x}'_i \beta)$  is relatively small. This is not surprising because asymptotically  $\hat{\beta} \xrightarrow{p} \beta$ , so  $\exp(\mathbf{x}'_i \hat{\beta}) \xrightarrow{p} \exp(\mathbf{x}'_i \beta)$ .

Much more difficult is using  $\exp(\mathbf{x}'_i \hat{\beta})$  to predict  $y_i|\mathbf{x}_i$  rather than  $E(y_i|\mathbf{x}_i)$  because there is always intrinsic randomness in  $y_i$ . In our example,  $y_i$  without any regression has a standard deviation of 7.95. Even if Poisson regression explains the data well enough to reduce the standard deviation of  $y_i|\mathbf{x}_i$  to, say, 4, then any prediction of  $y_i|\mathbf{x}_i$  will have a standard error of prediction of at least 4.

More generally, with microeconometric data and a large sample, we can predict the conditional mean  $E(y_i|\mathbf{x}_i)$  well but not  $y_i|\mathbf{x}_i$ . For example, we may predict well the mean earnings of a white female with 12 years of schooling but will predict relatively poorly the earnings of a randomly chosen white female with 12 years of schooling.

When the goal of prediction is to obtain a sample average predicted value, the sample average prediction should be a weighted average. To obtain a weighted average, specify weights with `summarize` or with `mean`; see section [3.8](#). This is especially important if one wants to make statements about the population and sampling is not simple random sampling. For average predictions, a simpler method is to use the `margins` command; see section [13.6](#).

### 13.5.3 Out-of-sample prediction

Out-of-sample prediction is possible. For example, we may want to make predictions for the 2001 sample using parameter estimates from the 2002 sample.

The dataset has data for both 2001 and 2002. We estimate using only 2002 data, then restrict the data in memory to 2001 data and predict for 2001 using the most recent coefficient estimates that were obtained using the 2002 data. We have

```
. * Out-of-sample prediction for year01 data using year02 estimates
. qui use mus210mepsdocvisyoung, clear
. qui poisson docvis private chronic female income if year02==1, vce(robust)
. keep if year01 == 1
(23,940 observations deleted)
. predict muhatyear01, n
. summarize docvis muhatyear01
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	6,184	3.896345	7.873603	0	152
muhatyear01	6,184	4.086984	2.963843	.7947512	15.02366

Note that the average of the predictions of  $E(y|\mathbf{x})$ , 4.09, no longer equals the average of the  $y$  values.

### 13.5.4 Prediction at a specified value of one of the regressors

Suppose we want to calculate the sample average number of doctor visits if all individuals had private insurance, whereas all other regressors are unchanged.

This can be done by setting `private = 1` and using `predict`. To return to the original data after doing so, we use the commands `preserve` to preserve the current dataset and `restore` to return to the preserved dataset. We have

```

. * Prediction at a particular value of one of the regressors
. qui use mus210mepsdocvisyoung, clear
. keep if year02 == 1
(25,712 observations deleted)
. qui poisson docvis private chronic female income, vce(robust)
. preserve
. replace private = 1
(947 real changes made)
. predict muhatpeq1, n
. summarize muhatpeq1

```

Variable	Obs	Mean	Std. dev.	Min	Max
muhatpeq1	4,412	4.371656	2.927381	1.766392	15.48004

```

. restore

```

The conditional mean is predicted to be 4.37 visits when all have private insurance, compared with 3.96 in the sample where only 78% had private insurance.

The preceding code calculates and stores the prediction for each individual. Usually, only the average of these predictions is required. Then it is much simpler to use the `margins` command, which also gives a 95% confidence interval for the average prediction; see section [13.6](#).

### 13.5.5 Prediction at a specified value of all the regressors

We may also want to estimate the conditional mean at a given value of all the regressors. For example, consider the number of doctor visits for a privately insured woman with no chronic conditions and an income of \$10,000.

To do so, we can use the `lincom` and `nlcom` commands. These commands compute point estimates for linear combinations and associated standard errors,  $z$  statistics,  $p$ -values, and confidence intervals. They are primarily intended to produce confidence intervals for parameter combinations such as  $\beta_3 - \beta_4$  and are presented in detail in sections [11.3.10](#) and [11.3.11](#). They can also be used for prediction because a prediction is a linear combination of the parameters.

We need to predict the number of doctor visits when `private = 1`, `chronic = 0`, `female = 1`, and `income = 10`. The `nlcom` command has the form

```
. * Predict at a specified value of all the regressors using nlcom
. nlcom exp(_b[_cons]+_b[private]*1+_b[chronic]*0+_b[female]*1+_b[income]*10)
    _nl_1: exp(_b[_cons]+_b[private]*1+_b[chronic]*0+_b[female]*
> 1+_b[income]*10)
```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
_nl_1	2.995338	.1837054	16.31	0.000	2.635282 3.355394

A simpler command for our example uses `lincom` with the `eform` option to display the exponential. Coefficients are then more simply referred to as `private`, for example, rather than `_b[private]`. We have

```
. * Predict at a specified value of all the regressors using lincom
. lincom _cons + private*1 + chronic*0 + female*1 + income*10, eform
( 1) [docvis]private + [docvis]female + 10*[docvis]income + [docvis]_cons = 0
```

docvis	exp(b)	Std. err.	z	P> z	[95% conf. interval]
(1)	2.995338	.1837054	17.89	0.000	2.656081 3.377929

The predicted conditional mean number of doctor visits is 3.00. The standard error of the prediction is 0.18, and a 95% confidence interval is [2.66, 3.38]. The standard error is computed with the delta method, and the bounds of the confidence interval depend on the standard error; see section [11.3.11](#). The test against a value of 0 is not relevant here but is relevant when `lincom` is used to test linear combinations of parameters.

The relatively tight confidence interval is for  $\exp(\mathbf{x}'\hat{\boldsymbol{\beta}})$  as an estimate of  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ . If instead we want to predict the actual values of  $y$  given  $\mathbf{x}$ , then the confidence interval will be much, much wider because we also need to add in variation in  $y$  around its conditional mean. There is considerable more noise in the prediction of the actual value than in estimating the conditional mean.

An even simpler method, using the `margins` command, is presented in section [13.6](#).

### 13.5.6 Prediction of other quantities

We have focused on prediction of the conditional mean. Options of the `predict` command provide prediction of other quantities of interest, where these quantities vary with the estimation command. Usually, one or more residuals are available. Following `poisson`, the `predict` option `score` computes the residual  $y_i - \exp(\mathbf{x}_i' \hat{\boldsymbol{\beta}})$ . An example of more command-specific predictions are those following the survival data command `streg` to produce not only mean survival time but also median survival time, the hazard, and the relative hazard.

For a discrete dependent variable, it can be of interest to obtain the predicted probability of each of the discrete values; that is,  $\Pr(y_i = 0)$ ,  $\Pr(y_i = 1)$ ,  $\Pr(y_i = 2)$ , .... For binary logit and probit, the default `pr()` option of `predict` gives  $\Pr(y_i = 1)$ . The `pr()` option is available for many fully parametric models and gives ranges of probabilities as well as, for discrete outcomes, the probability of a particular value. For the use of `predict` in multinomial models, see section 18.4.

## 13.6 Predictive margins

Predictive margins for the conditional mean  $\mathbf{x}'\boldsymbol{\beta}$  following OLS regression using the `regress` command were presented in some detail in section [4.4](#). That section distinguished between sample-average margins and margins at particular values of regressor variables, plots using `marginsplot`, pairwise comparisons of predictive margins using `margins`, `pwcompare`, and multiple comparisons using `margins`, `contrast`.

That material is all relevant for nonlinear models. We give a much shorter presentation here that focuses on the different quantities that are predicted in a nonlinear model. For example, following Poisson regression, interest lies in the conditional mean  $\exp(\mathbf{x}'\boldsymbol{\beta})$ .

### 13.6.1 The `margins` command

The `margins` command simplifies prediction at specified values of the regressors. It predicts

$$PM = \sum_{i=1}^N g\left(\mathbf{x}_i^*, \hat{\boldsymbol{\beta}}\right)$$

Different options of the command use different functions  $g(\cdot)$  and evaluate at different values  $\mathbf{x}_i^*$ ,  $i = 1, \dots, N$ .

The syntax of the command is

```
margins [marginlist] [if] [in] [weight] [, response_options options]
```

where *marginlist* is a list of variable names or of factor variables that appear in the current estimation results that are being analyzed, *response\_options* specify the particular quantity to be computed, and *options* include particular values of regressors at which computation occurs.

A key *response\_option* is `predict()`, which defines the quantity predicted. For example, following the `poisson` command, the `margins` command can be used to compute the number of events (default option `predict(n)`), the incident rate (option `predict(ir)`), the linear prediction (option `predict(xb)`), and probabilities (option `predict(pr())`).

The *response\_option* `expression()` allows the user to define the formula of interest. For example, `margins, expression(exp(predict(xb)))` following the `poisson` command provides predictive margins for  $\exp(\mathbf{x}'\boldsymbol{\beta})$ .

The *response\_options* `dydx()`, `eyex()`, `dyex()`, and `eydx()` calculate MES presented in section [13.7](#). Key *options* of the `margins` command are the `at()` and `atmeans` options illustrated below. The default computes the sample average value of the default `margins` prediction.

### 13.6.2 Predictive margins: Average, at specified values, and at mean

Following the `poisson` command, therefore, the `margins` command yields the sample average of the predicted number of events. We have

```
. * margins: Sample average of predicted number of events
. qui poisson docvis private chronic female income, vce(robust)
. margins
Predictive margins                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()


```

---

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	3.957389	.1115373	35.48	0.000	3.73878	4.175998

A 95% confidence interval for the sample average predicted number of doctor visits is [3.74, 4.18].

To predict at specified values of one or more regressors, we use the `at()` option.

For example, the average number of doctor visits if all individuals had private insurance, with other regressors equal to sample values, is calculated

using the following command:

```
. * margins: Sample average prediction at a particular value of a regressor
. margins, at(private=1)

Predictive margins                                         Number of obs = 4,412
Model VCE: Robust

Expression: Predicted number of events, predict()
At: private = 1
```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	4.371656	.1273427	34.33	0.000	4.122069	4.621243

The average equals that given in section [13.5.4](#), with 95% confidence interval [4.12, 4.62].

The `at()` option can be used to compute the predicted number of doctor visits at specified values of all the regressors. For example,

```
. * margins: Prediction at a specified value of all regressors
. margins, at(private=1 chronic=0 female=1 income=10)

Adjusted predictions                                         Number of obs = 4,412
Model VCE: Robust

Expression: Predicted number of events, predict()
At: private = 1
    chronic = 0
    female = 1
    income = 10
```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	2.995338	.1837054	16.31	0.000	2.635282	3.355394

The results coincide with those given in section [13.5.5](#).

The `atmeans` option computes the predicted number of doctor visits when regressors are evaluated at the sample mean of regressors. From output not given, the `margins, atmeans` command yields  $\exp(\bar{\mathbf{x}}'\hat{\boldsymbol{\beta}})$  equal to 3.02968. This is different from  $1/N \sum_i \exp(\mathbf{x}_i'\hat{\boldsymbol{\beta}})$ , the average of the predicted number of doctor visits, which from previous output equals

3.957389. The average of a nonlinear function does not equal the nonlinear function evaluated at the average.

### 13.6.3 Predictive margins for a categorical factor variable

When the estimation command variable list includes a categorical factor variable, we can easily obtain average predictions for each value of the factor variable. For example, for private insurance, we obtain

```
. * margins: Sample avg. prediction at different values of an indicator variable
. qui poisson docvis i.private chronic female income, vce(robust)
. margins private
Predictive margins                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
private						
0	1.966935	.2054776	9.57	0.000	1.564207	2.369664
1	4.371656	.1273427	34.33	0.000	4.122069	4.621243

The `i.` operator is used in the estimation command to signal that variable `private` is a categorical variable. If the categorical variable instead took four values, say, then estimation would include three indicator variables (with a base category omitted) and the `margins` command would compute the average prediction for each of the four values of the categorical variable.

## 13.7 Marginal effects

An ME, or partial effect, most often measures the effect on the conditional mean of  $y$  of a change in one of the regressors, say,  $x_j$ . In the linear regression model, the ME equals the relevant slope coefficient, greatly simplifying analysis. For nonlinear models, this is no longer the case, leading to remarkably many different methods for calculating MES. Also, other MES may be desired, such as elasticities and effects on conditional probabilities rather than the conditional mean. A different type of ME, the marginal treatment effects (MTES), is presented at the end of this section.

The presentation here extends the methods presented in detail in section [4.5](#) for linear regression to nonlinear models and is more detailed than the introductory treatment for nonlinear models in section [10.4](#).

### 13.7.1 Calculus and finite-difference methods

Calculus methods can be applied for a continuous regressor, and the ME of the  $j$ th regressor is then

$$\text{ME}_j = \frac{\partial E(y|\mathbf{x} = \mathbf{x}^*)}{\partial x_j}$$

For the Poisson model with  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ , we obtain  $\text{ME}_j = \exp(\mathbf{x}^{*\prime}\boldsymbol{\beta})\beta_j$ . This ME is not simply the relevant parameter  $\beta_j$ , and it varies with the point of evaluation  $\mathbf{x}^*$ .

Calculus methods are not always appropriate. In particular, for an indicator variable, say,  $d$ , the relevant ME is the change in the conditional mean when  $d$  changes from 0 to 1. Let  $\mathbf{x} = (\mathbf{z} \ d)$ , where  $\mathbf{z}$  denotes all regressors other than the  $j$ th, which is an indicator variable  $d$ . Then the finite-difference method yields the ME

$$\text{ME}_j = E(y|\mathbf{z} = \mathbf{z}^*, d = 1) - E(y|\mathbf{z} = \mathbf{z}^*, d = 0)$$

Even for continuous regressors, we may want to consider discrete changes, such as the impact of age increasing from 40 to 60. Letting  $\mathbf{x} = (\mathbf{z} \ w)$ , the finite-difference method uses

$$\text{ME}_j = E(y|\mathbf{z} = \mathbf{z}^*, w = 60) - E(y|\mathbf{z} = \mathbf{z}^*, w = 40)$$

The `contrast` and `margins`, `contrast` commands use the finite-difference method.

For the linear regression model, with all regressors entering linearly rather than interactively, calculus and finite-difference methods give the same result. For nonlinear models, this is no longer the case. If interest lies in a unit change in the regressor then in nonlinear models calculus methods only provide an approximation.

Finally, as for linear models, interactions in regressors lead to additional complications.

### 13.7.2 Average marginal effect, ME at mean, and ME at a representative value

For nonlinear models, the ME varies with the point of evaluation. Three common choices of evaluation are 1) at sample values and then average, 2) at the sample mean of the regressors, and 3) at representative values of the regressors. We use the following acronyms, where the first two follow [Bartus \(2005\)](#):

AME	AME	Average of ME at each $\mathbf{x} = \mathbf{x}_i$
MEM	Marginal effect at mean	ME at $\mathbf{x} = \bar{\mathbf{x}}$
MER	Marginal effect at a representative value	ME at $\mathbf{x} = \mathbf{x}^*$

In a nonlinear model, these provide different ME estimates. The in-sample AME is often used because it provides a simple interpretation of results

obtained from a nonlinear model when direct interpretation of the coefficients can be difficult. The MEM is used if interest lies in the ME for the average individual in the sample, while the MER is used if interest lies in the ME for an individual with a specific set of characteristics.

The three quantities can be computed using the `margins` postestimation command with the `dydx()` option. Combinations of these methods are also possible. For example, several regressors may be set to a representative value, while the remaining regressors are evaluated at sample values and then the average is taken.

### 13.7.3 Simple interpretations of coefficients in single-index models

In nonlinear models, coefficients are more difficult to interpret because now  $\beta_j \neq \partial E(y|\mathbf{x})/\partial x_j$ . Nonetheless, some direct interpretation is possible if the conditional mean is of the single-index form

$$E(y|\mathbf{x}) = m(\mathbf{x}'\boldsymbol{\beta})$$

This single-index form implies that the ME is

$$\text{ME}_j = m'(\mathbf{x}'\boldsymbol{\beta}) \times \beta_j$$

where  $m'(\mathbf{x}'\boldsymbol{\beta})$  denotes the derivative of  $m(\mathbf{x}'\boldsymbol{\beta})$  with respect to  $\mathbf{x}'\boldsymbol{\beta}$ .

Two important properties follow. First, if  $m(\mathbf{x}'\boldsymbol{\beta})$  is monotonically increasing, so  $m'(\mathbf{x}'\boldsymbol{\beta}) > 0$  always, then the sign of  $\hat{\beta}_j$  gives the sign of the ME [and if  $m(\mathbf{x}'\boldsymbol{\beta})$  is monotonically decreasing, then sign of  $\hat{\beta}_j$  is the negative of that of the ME]. Second, for any function  $m(\cdot)$  and at any value of  $\mathbf{x}$ , we have

$$\frac{\text{ME}_j}{\text{ME}_k} = \frac{\beta_j}{\beta_k}$$

Therefore, if one coefficient is twice as big as another, then so too is the ME. These two properties apply to most commonly used nonlinear regression models, aside from multinomial models.

For example, from section [13.3.2](#), the regressor `private` has a coefficient of 0.80, and the regressor `chronic` has a coefficient of 1.09. It follows that having a chronic condition is associated with a bigger change in doctor visits than having private insurance because  $1.09 > 0.80$ . The effects for both regressors are positive because the coefficients are positive and the conditional mean  $\exp(\mathbf{x}'\boldsymbol{\beta})$  is a monotonically increasing function.

Additional interpretation of coefficients can be possible for specific single-index models. In particular, for the exponential conditional mean  $\exp(\mathbf{x}'\boldsymbol{\beta})$ , the  $\text{ME}_j = E(y|\mathbf{x}) \times \beta_j$ . So  $\beta_j = \text{ME}_j/E(y|\mathbf{x})$ , and from [\(13.12\)](#) the regression coefficients can be interpreted as semielasticities. From section [13.3.2](#), the regressor `income` has a coefficient of 0.0036. It follows that a \$1,000 increase in income (a one-unit increase in the rescaled regressor `income`) is associated with a 0.0036 proportionate increase, or a 0.36% increase, in doctor visits.

Using instead the finite-difference method, we see a one-unit change in  $x_j$  that implies that  $\mathbf{x}'\boldsymbol{\beta}$  changes to  $\mathbf{x}'\boldsymbol{\beta} + \beta_j$ , so  $\text{ME}_j = \exp(\mathbf{x}'\boldsymbol{\beta} + \beta_j) - \exp(\mathbf{x}'\boldsymbol{\beta}) = (e^{\beta_j} - 1)\exp(\mathbf{x}'\boldsymbol{\beta})$ . This is a proportionate increase of  $(e^{\beta_j} - 1)$ , or a percentage change of  $100 \times (e^{\beta_j} - 1)$ .

### 13.7.4 The `dydx()` option of the `margins` command for MEs

The `margins` postestimation command is detailed in section [13.6.1](#). The `dydx()` option computes MEs. The variant `dydx(*)` computes the ME for all regressors. Alternatively, a subset of regressors can be explicitly listed in the parentheses.

The default is to compute the AME. Alternatively, the MEM can be computed using the `atmeans` option, and the MER can be computed using the `at()` option.

All of these MES are, by default, calculated using calculus methods. To instead use the finite-difference method for binary regressors, one must use the `i.` operator in the preceding estimation command to signal that the regressor is a binary variable. This additional step is strongly preferred because the change of interest is from one value of the binary variable to the other, whereas calculus methods consider an infinitesimally small change whose effect is then scaled up to correspond to a large change. The two estimates differ in a nonlinear model.

The default standard errors treat the regressors as fixed. If a population ME is desired and sample values of regressors are used in computing the ME, then the option `vce(unconditional)` additionally controls for variation in the regressors because of sampling; see section [13.7.9](#).

The default MES are computed for the default prediction for the `predict` command. For many nonlinear models, such as the Poisson count model, the default MES are therefore computed for  $E(y|x)$ . For multinomial models, such as multinomial logit, the default MES are instead computed for  $\Pr(y = j|x)$ ,  $j = 1, \dots, m$ , that is, the probability for each of the  $m$  outcomes. For models with censoring or selection, such as the `tobit` or `heckman` command, the default MES are computed for the index  $x'\beta$  so that the default ME is simply the estimated regression coefficient.

MES for some other quantities can be computed using the `predict(predict_option)` option of the `margins` command, where the specific `predict_option` varies with the preceding estimation command. And the `expression()` option (see section [13.6.1](#)) can be used to define a quantity of interest. Section [13.7.8](#) provides an ME example.

### 13.7.5 Average marginal effect

The `margins` command with just the `dydx()` option yields the AMES for the default option of the `predict` command. After `poisson`, the default prediction is one for  $E[y|x]$ .

The default is to use calculus methods to compute the ME. For the three binary regressors, we instead use the finite-difference method by using the

i. operator in the `poisson` estimation command before applying the `margins` command.

For the doctor-visits data, with the `i.` operator used in estimation so that the AMES for binary regressors are computed using the finite-difference method, we obtain the following:

```
. * AMEs using margins command and finite differences
. qui poisson docvis i.private i.chronic i.female income, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 4,412
Model VCE: Robust

Expression: Predicted number of events, predict()
dy/dx wrt: 1.private 1.chronic 1.female income
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	2.404721	.2438573	9.86	0.000	1.926769	2.882672
1.chronic	4.599174	.2886176	15.94	0.000	4.033494	5.164854
1.female	1.900212	.2156694	8.81	0.000	1.477508	2.322917
income	.0140765	.0043457	3.24	0.001	.0055591	.0225939

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

In this example, the AMES are about three times the estimated Poisson coefficients.

When we average across individuals, people with private insurance have on average 2.40 more doctor visits than those without private insurance after controlling for income, gender, and chronic conditions.

The AMES for single-index models, such as the Poisson model, are in practice quite similar to the coefficients obtained by OLS regression of  $y$  on  $x$ . This is indeed the case here because the OLS coefficients, from output not given, are 1.92, 4.82, 1.89, and 0.016.

#### Comparison of calculus and finite-difference methods

To show that calculus and finite-difference methods can differ considerably, we repeat the command following the `poisson` command without using the

i. operator to signal which regressors are binary variables. Then calculus methods are used for all variables. We obtain

```
. * AMEs using margins command and only calculus method
. qui poisson docvis private chronic female income, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 4,412
Model VCE: Robust

Expression: Predicted number of events, predict()
dy/dx wrt: private chronic female income
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
private	3.160629	.4352572	7.26	0.000	2.307541	4.013717
chronic	4.320935	.2757872	15.67	0.000	3.780402	4.861468
female	1.949204	.2313726	8.42	0.000	1.495722	2.402686
income	.0140765	.0043457	3.24	0.001	.0055591	.0225939

The AMEs for the three binary regressors change, respectively, from 2.40 to 3.16, 4.60 to 4.32, and 1.90 to 1.95, while the AME for the continuous regressor `income` is unchanged. For binary regressors, the first method, using finite differences, is conceptually better.

### 13.7.6 Marginal effect at the mean

The `margins` command with the `dydx()` and `atmeans` options yields the ME evaluated at the mean for the default option of the `predict` command. After `poisson`, the default prediction is one for  $E(y|x)$ , so the MEs give the change in the expected number of doctor visits when the regressor changes, evaluated at  $x = \bar{x}$ .

We have

```

. * MEMs using margins command and finite differences
. qui poisson docvis i.private i.chronic i.female income, vce(robust)
. margins, dydx(*) atmeans
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: 1.private 1.chronic 1.female income
At: 0.private = .2146419 (mean)
     1.private = .7853581 (mean)
     0.chronic = .6736174 (mean)
     1.chronic = .3263826 (mean)
     0.female = .5281052 (mean)
     1.female = .4718948 (mean)
     income = 34.34018 (mean)

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	1.978178	.204407	9.68	0.000	1.577547	2.378808
1.chronic	4.200068	.2794096	15.03	0.000	3.652435	4.7477
1.female	1.528406	.1775762	8.61	0.000	1.180363	1.876449
income	.0107766	.0033149	3.25	0.001	.0042796	.0172737

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

The output includes the regressor values at which the MES are calculated. In subsequent examples, we use the `noatlegend` option to suppress this part of the output.

For the average individual, where average means evaluation at the sample means of the regressors, the number of doctor visits increases by 1.98 for those with private insurance and by 0.011 with a \$1,000 increase in annual income.

For nonlinear models, average behavior of individuals differs from behavior of the average individual. The direction of the difference is generally indeterminate, but for models with an exponential conditional mean, it can be shown that the `atmeans` option (giving the MEM) will produce smaller MES than the default (giving the AME), a consequence of the exponential function being globally convex.

Thus, in this example, the MEMs are, respectively, 1.98, 4.20, 1.53, and 0.011, while the AMEs are, respectively, 2.40, 4.60, 1.90, and 0.014.

### 13.7.7 Marginal effect at a representative value

The `margins` command with the `dydx()` and `at()` options yields the ME evaluated at a particular value of the regressors.

As an example, we consider computing the MES for a privately insured woman with no chronic conditions and income of \$10,000. We use the `i.` operator in estimation so that the MES for binary regressors are computed using the finite-difference method. We have

```
. * MERs using margins command
. qui poisson docvis i.private i.chronic i.female income, vce(robust)
. margins, dydx(*) at(private=1 chronic=0 female=1 income=10) noatlegend
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: 1.private 1.chronic 1.female income
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	1.647648	.2072813	7.95	0.000	1.241385	2.053912
1.chronic	5.930251	.4017655	14.76	0.000	5.142805	6.717697
1.female	1.164985	.1546063	7.54	0.000	.8619621	1.468008
income	.0106545	.0028688	3.71	0.000	.0050317	.0162772

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

For example, having private insurance is estimated to increase the number of doctor visits by 1.65, with 95% confidence interval [1.24, 2.05]. Note that the confidence interval is for the change in the expected number of visits  $E(y|x)$ . A confidence interval for the change in the actual number of visits ( $y|x$ ) will be much wider; see the analogous discussion for prediction in section [13.5.2](#).

Related `at()` options allow evaluation at a mix of user-provided values and sample means of regressors.

### 13.7.8 MEs with respect to a user-defined quantity

The `expression()` option of the `margins` command can be used to obtain MES with respect to a user-defined quantity. For example, following Poisson

regression, the preceding default AMEs for the conditional mean can equivalently be obtained using the command

```
. * AME semielasticity evaluated at mean of regressors  
. margins, dydx(*) expression(exp(predict(xb)))  
(output omitted)
```

Here `predict(xb)` following Poisson regression gives  $\mathbf{x}'\boldsymbol{\beta}$ , and exponentiating then gives the conditional mean.

### 13.7.9 Which ME to use?

The three MES—AME, MEM, and MER—can differ appreciably in nonlinear models. Which ME should be used? It is common to use the AME in reporting regression results. For policy analysis, one may want to instead obtain the MER for targeted values of the regressors by using the `at()` option of the `margins` command or the MEM for the average individual by using the `atmeans` option.

The AME is most often reported as an in-sample AME. If instead this AME is to be interpreted as a population AME, then standard errors for the AME should be computed using the `vce(unconditional)` option of the `margins` command because it additionally controls for variation in the regressors due to sampling. This is appropriate if interest lies in extrapolating the impact of a policy impact measured in one setting, such as a particular school district, to another school district drawn from a population of similar school districts. There are then two sources of randomness in computing MEs—randomness in the parameter estimates and randomness in the regressors. The first of these is usually much larger; in the preceding AME example, adding `vce(unconditional)` leads to an increase in AME standard errors of less than 1%.

Microeconometric studies often use survey data that are stratified on exogenous regressors, in which case the estimators are consistent but the regressors  $\mathbf{x}$  may be unrepresentative of the population. For example, a nonlinear regression of earnings ( $y$ ) on schooling ( $x$ ) may use a dataset for which individuals with low levels of schooling are oversampled, so that the sample mean  $\bar{x}$  is less than the population mean. Then MER can be used with no modification. But MEM and AME should be evaluated by using sampling

weights, introduced in section 3.8. In particular, the AME should be computed as a weighted average, using sampling weights, of the ME for each individual. Estimation itself is most often unweighted, though if weights are used in estimation, then calculations in `margins` automatically adjust for any weights used during estimation; this can be changed by using the `noweights` option of the `margins` command.

### 13.7.10 AME computed manually

The AME can always be computed manually using the following method. Predict at the current sample values for all observations, change one regressor by a small amount, predict at the new values, subtract the two predictions, and divide by the amount of the change. The AME is the average of this quantity. By choosing a very small change, we replicate the calculus method. A finite-difference estimate is obtained by considering a large change such as a one-unit change (whether this is large depends on the scaling of the regressors). In either case, we use the `preserve` and `restore` commands to return to the original data after computing the AME.

We consider the effect of a small change in income on doctor visits. This yields a crude approximation to the derivative. For more precise numerical estimates of the derivative, see section 5.7 of [Press et al. \(2007\)](#). We have

```
. * AME computed manually for a single regressor
. qui use mus210mepsdocvisyoung, clear
. keep if year02 == 1
(25,712 observations deleted)
. qui poisson docvis private chronic female income, vce(robust)
. preserve
. predict mu0, n
. qui replace income = income + 0.01
. predict mu1, n
. generate memanual = (mu1-mu0)/0.01
. summarize memanual
      Variable |       Obs        Mean    Std. dev.       Min       Max
                 |   4,412     .0140761     .0106173     .0028253     .055027
. restore
```

The AME estimate is 0.0140761, essentially the same as the 0.0140765 obtained by using `margins` in section [13.7.5](#). This method gives no standard error for the AME estimate. Instead, it computes the standard deviation of the AME for the 4,412 observations.

A better procedure chooses a change that varies with the scaling of each regressor. We use a change equal to the standard deviation of the regressor divided by 1,000. We also use the looping command `foreach` to obtain the AME for each variable. We have the following:

```

. * AME computed manually for all regressors
. global xlist private chronic female income
. preserve
. predict mu0, n
. foreach var of varlist $xlist {
2.     qui summarize `var'
3.     generate delta = r(sd)/1000
4.     qui generate orig = `var'
5.     qui replace `var' = `var' + delta
6.     predict mu1, n
7.     qui generate me_`var' = (mu1 - mu0)/delta
8.     qui replace `var' = orig
9.     drop mu1 delta orig
10. }
. summarize me_*

```

Variable	Obs	Mean	Std. dev.	Min	Max
me_private	4,412	3.16153	2.384785	.6349193	12.36743
me_chronic	4,412	4.322181	3.260333	.8679963	16.90794
me_female	4,412	1.949399	1.470413	.3915812	7.625329
me_income	4,412	.0140772	.0106184	.0028284	.055073

```

. restore

```

The AME estimate for `income` is the average 0.0140772, essentially the same as the 0.0140765 produced by using the `margins, dydx()` command. The other AMEs differ from those given in section [13.7.5](#) because here we used calculus methods, whereas in section [13.7.5](#), the AMEs for binary regressors were computed using the finite-difference method.

The code can clearly be adapted to use finite differences in those cases. In nonstandard models, such as those fit by using the `ml` command, it will be necessary to also provide code to replace the `predict` command.

Note that the reported standard deviation, such as 0.0106184 for `me_income`, is the standard deviation across individuals of the estimated ME. To compute the standard error of the AME, one then needs to embed the preceding code in a bootstrap loop.

### 13.7.11 Polynomial regressors

Regressors may appear as polynomials. Then computing MES becomes considerably more complicated.

First, consider a linear model that includes a cubic function in regressor  $z$ . Then,

$$\begin{aligned} E(y|\mathbf{x}, z) &= \mathbf{x}'\boldsymbol{\beta} + \alpha_1 z + \alpha_2 z^2 + \alpha_3 z^3 \\ \Rightarrow \quad \text{ME}_z &= \alpha_1 + 2\alpha_2 z + 3\alpha_3 z^2 \end{aligned}$$

The AME can be computed by calculating  $\hat{\alpha}_1 + 2\hat{\alpha}_1 z + 3\hat{\alpha}_1 z^2$  for each observation and averaging.

We do so for a slightly more difficult example, the Poisson model. Then,

$$\begin{aligned} E(y|\mathbf{x}, z) &= \exp(\mathbf{x}'\boldsymbol{\beta} + \alpha_1 z + \alpha_2 z^2 + \alpha_3 z^3) \\ \Rightarrow \quad \text{ME}_z &= \exp(\mathbf{x}'\boldsymbol{\beta} + \alpha_1 z + \alpha_2 z^2 + \alpha_3 z^3) \times (\alpha_1 + 2\alpha_2 z + 3\alpha_3 z^2) \end{aligned}$$

A cubic function in income is part of our model for doctor visits. Below, we estimate the parameters of the model by ML.

```

. * AME for a polynomial regressor: Manual computation
. generate inc2 = income^2
. generate inc3 = income^3
. qui poisson docvis private chronic female income inc2 inc3, vce(robust)
. predict muhat, n
. generate me_income = muhat*(_b[income]+2*_b[inc2]*income+3*_b[inc3]*inc2)
. summarize me_income

```

Variable	Obs	Mean	Std. dev.	Min	Max
me_income	4,412	.0178233	.0137618	-.0534614	.0483436

The code uses the simplification that ME<sub>z</sub> =  $E(y|\mathbf{x}, z) \times (\alpha_1 + 2\alpha_2 z + 3\alpha_3 z^2)$ . The average of the individual MEs of a change in income is 0.0178 in the cubic model, compared with 0.0141 when income enters only linearly.

This AME can be more simply computed using factor variables. From section [1.3.4](#), a polynomial in variable `income` can be specified using the `c.` operator (for a continuous variable) and the `#` operator for interaction. The ME of `income` can then be computed using the usual `margins` command. We have

```

. * AME for a polynomial regressor: Computation using factor variables
. qui poisson docvis private chronic female c.income c.income#c.income
>      c.income#c.income#c.income, vce(robust)
. margins, dydx(income)

Average marginal effects                                         Number of obs = 4,412
Model VCE: Robust

Expression: Predicted number of events, predict()
dy/dx wrt: income

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
income	.0178233	.0062332	2.86	0.004	.0056064	.0300402

The AME equals that computed manually, but additionally, we have a 95% confidence interval of [0.0056, 0.0300]. The MEM and MER can also be computed with the `atmeans` and `at()` options.

### 13.7.12 Interacted regressors

Similar issues arise with regressors interacted with indicator variables. For example,

$$E(y|\mathbf{x}, z, d) = \exp(\mathbf{x}'\boldsymbol{\beta} + \alpha_1 z + \alpha_2 d + \alpha_3 d \times z)$$

$$\Rightarrow ME_z = \exp(\mathbf{x}'\boldsymbol{\beta} + \alpha_1 z + \alpha_2 d + \alpha_3 d \times z) \times (\alpha_1 + \alpha_3 d)$$

[Long and Freese \(2014\)](#) give an extensive discussion of MES with interactive regressors, performed by using the community-contributed prvalue command presented in section 17.5.7. These are oriented toward calculation of the MEM or MER rather than the AME.

Factor variables automate the computation of MES in models with interactions. As an example, we modify the doctor-visits model to include an interaction between the binary regressor `female` and the continuous regressor `income`.

We first need to specify the model fit with factor variables, using the relevant `c.`, `i.`, and `#` operators. We have

```
. * Specify model with interacted regressors using factor variables
. poisson docvis private chronic i.female c.income i.female#c.income,
> vce(robust) nolog
Poisson regression
Number of obs = 4,412
Wald chi2(5) = 606.43
Prob > chi2 = 0.0000
Pseudo R2 = 0.1943
Log pseudolikelihood = -18475.536
```

docvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	.802035	.1084187	7.40	0.000	.5895383	1.014532
chronic	1.094331	.0563504	19.42	0.000	.9838865	1.204776
1.female	.6328542	.0927712	6.82	0.000	.451026	.8146823
income	.0051734	.0015708	3.29	0.001	.0020946	.0082522
female#c.income						
1	-.0035176	.0019089	-1.84	0.065	-.0072589	.0002237
_cons	-.3082426	.1242329	-2.48	0.013	-.5517346	-.0647506

The coefficient of `female` is labeled `1.female`, and the coefficient of `female × income` is labeled `1.female#c.income`.

We then compute the AME of variables `female` and `income`. We have

```
. * AME with interacted regressors given model specified using factor variables
. margins, dydx(female income)
Average marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: 1.female income
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
1.female	1.884021	.2159008	8.73	0.000	1.460864
income	.0116832	.0037989	3.08	0.002	.0042375

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

The average of the individual MES of a change in income is 0.0117 compared with 0.0178 in the cubic model and 0.0141 when income enters only linearly. The AME of moving from male (`female` = 0) to female (`female` = 1) is 1.884 office visits, compared with 1.900 from section [13.7.5](#) where there was no interaction. The output includes 95% confidence intervals for the AMES. The MEMS and MERS can alternatively be computed, using the `atmeans` and `at()` options of the `margins` command.

### 13.7.13 Complex interactions and nonlinearities

MES in models with interactions can become very difficult to interpret and calculate.

For complex interactions, a simple procedure is to compute the ME by manually changing the relevant variables and interactions, recomputing the predicted conditional mean, and subtracting. We change by an amount  $\Delta$ , a single variable that, because of interactions or polynomials, or both, appears several times as a regressor. Let the original values of  $\mathbf{x}_i$  be denoted by  $\mathbf{x}_{i0}$ , and obtain the prediction  $\hat{\mu}_{i0} = \exp(\mathbf{x}'_{i0}\hat{\beta})$ . Then, change the variable by  $\Delta$  to give new values of  $\mathbf{x}_i$  denoted by  $\mathbf{x}_{i1}$ , and obtain the prediction  $\hat{\mu}_{i1} = \exp(\mathbf{x}'_{i1}\hat{\beta})$ . Then, the ME of changing the variable is  $(\hat{\mu}_{i1} - \hat{\mu}_{i0})/\Delta$ .

We illustrate this for the cubic in income example.

```

. * AME computed manually for a complex model
. preserve
. predict mu0, n
. qui summarize income
. generate delta = r(sd)/100
. qui replace income = income + delta
. qui replace inc2 = income^2
. qui replace inc3 = income^3
. predict mu1, n
. generate me_inc = (mu1 - mu0)/delta
. summarize me_inc

```

Variable	Obs	Mean	Std. dev.	Min	Max
me_inc	4,412	.0116899	.0089482	.0022914	.1083069

```
. restore
```

This reproduces the calculus result because it considers a small change in income, here one-hundredth of the standard deviation of income. If instead we had used `delta=1`, then this program would have given the ME of a one-unit change in the regressor `income` (here a \$1,000 change) computed by using the finite-difference method.

For complex interactions, one can use factor variables, as shown in section [13.7.12](#), although for complex interactions, care is needed to correctly specify the factor variables and their interactions in the initial model estimation command.

### 13.7.14 Elasticities and semielasticities

The impact of changes in a regressor on the dependent variable can also be measured by using elasticities and semielasticities.

For simplicity, we consider a scalar regressor  $x$ , and the effect of a change in  $x$  on  $E(y|x)$ , which we write more simply as  $y$ . Then the ME using the finite-difference method is given by

$$\text{ME} = \frac{\Delta y}{\Delta x}$$

This measures the change in  $y$  associated with a one-unit change in  $x$ . We present elasticities based on finite differences. If instead calculus methods are used, we replace  $\Delta y/\Delta x$  in the equations below with the derivative  $\partial y/\partial x$ .

An elasticity instead measures the proportionate change in  $y$  associated with a given proportionate change in  $x$ . More formally, the elasticity  $\varepsilon$  is given by

$$\varepsilon = \frac{\Delta y/y}{\Delta x/x} = \frac{\Delta y}{\Delta x} \times \frac{x}{y} = \text{ME} \times \frac{x}{y} \quad (13.11)$$

For example, if  $y = 1 + 2x$ , then  $\Delta y/\Delta x = 2$  and the elasticity at  $x = 3$  equals  $2 \times 3/7 = 6/7 = 0.86$ . This can be interpreted as follows: a 1% increase in  $x$  is associated with a 0.86% increase in  $y$ .

Elasticities can be more useful than MES because they are scale-free measures. For example, suppose we estimate that a \$1,000 increase in annual income is associated with 0.1 more doctor visits per year. Whether this is a large or small effect depends on whether these changes in income and doctor visits are large or small. Given knowledge that the sample means of income and doctor visits are, respectively, \$34,000 and 4, the elasticity  $\varepsilon = 0.1 \times 34/4 = 0.85$ . This is a large effect. For example, a 10% increase in income is associated with an 8.5% increase in doctor visits.

A semielasticity is a hybrid of an ME and an elasticity that measures the proportionate change in  $y$  associated with a one-unit change in  $x$ . The semielasticity is given by

$$\frac{\Delta y/y}{\Delta x} = \frac{\Delta y}{\Delta x} \times \frac{1}{y} = \text{ME} \times \frac{1}{y} \quad (13.12)$$

For the preceding example, the semielasticity is  $0.1/4 = 0.025$ , so a \$1,000 increase in income (a one-unit change given that income is measured in

thousands of dollars) is associated with a 0.025 proportionate increase, or a 2.5% increase, in doctor visits.

Less used is the unit change in  $y$  associated with a proportionate change in  $x$ , given by

$$\frac{\Delta y}{\Delta x/x} = \frac{\Delta y}{\Delta x} \times x = \text{ME} \times x$$

These four quantities can be computed by using various options of the `margins` command, given in section [13.7.4](#). An illustration is given in section [13.7.15](#).

### 13.7.15 Elasticities and semielasticities example

The elasticities and semielasticities defined in section [13.7.14](#) can be computed by using `margins`'s options `eyex()`, `eydx()`, and `dyex()`. Evaluation can be at each sample value and then averaged (the default option), at the mean value of regressors (the `atmeans` option), or at a representative value of the regressors (the `at()` option).

We continue with the same Poisson regression example, with four regressors, but we focus on the impact of just the regressor `income`. For illustration, we evaluate elasticities and semielasticities at the mean value of regressors by using the `atmeans` option.

We first obtain the ME with the `dydx()` option.

```

. * Usual ME evaluated at means of regressors
. qui poisson docvis private chronic female income, vce(robust)
. margins, dydx(income) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: income

```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
income	.0107766	.0033149	3.25	0.001	.0042796 .0172737

For the average individual, the number of doctor visits increases by 0.0108 with a \$1,000 increase in annual income. This repeats the result given in section [13.7.6](#).

We next compute the elasticity. The `eyex()` option yields

```

. * Elasticity evaluated at means of regressors
. margins, eyex(income) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
ey/ex wrt: income

```

	Delta-method				
	ey/ex	std. err.	z	P> z	[95% conf. interval]
income	.1221485	.0371729	3.29	0.001	.049291 .195006

The elasticity is 0.122, so for the average individual, a 1% increase in income is associated with a 0.122% increase in doctor visits, or a 10% increase in income is associated with a 1.22% increase in doctor visits. The elasticity equals  $ME \times x/y$  from [\(13.11\)](#), where evaluation here is at  $x = 34.34$  (the sample mean of `income`) and  $y = 3.03$  (the predicted number of doctor visits for  $x = \bar{x}$  computed using the `margins, atmeans` command). This yields  $0.0108 \times 34.34/3.03 = 0.122$  as given in the above output.

The semielasticity is obtained with the `eydx()` option:

```

. * Semielasticity evaluated at means of regressors
. margins, eydx(income) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
ey/dx wrt: income

```

	Delta-method					
	ey/dx	std. err.	z	P> z	[95% conf. interval]	
income	.003557	.0010825	3.29	0.001	.0014354	.0056787

For the average individual, a \$1,000 increase in annual income (a one-unit change in `income`) is associated with a 0.003557 proportionate rise, or a 0.3557% increase in the number of doctor visits. This exactly equals the coefficient of `income` in the original Poisson regression (see section [13.3.2](#)), confirming that if the conditional mean is of exponential form, then the coefficient  $\hat{\beta}_j$  is already a semielasticity, as explained in section [13.7.3](#).

Finally, the `dyex()` option yields

```

. * Other semielasticity evaluated at means of regressors
. margins, dyex(income) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 4,412
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/ex wrt: income

```

	Delta-method					
	dy/ex	std. err.	z	P> z	[95% conf. interval]	
income	.3700708	.1138338	3.25	0.001	.1469607	.5931809

A proportionate increase of one in `income` (a doubling of `income`) is associated with 0.37 more doctor visits. Equivalently, a 1% increase in `income` is associated with 0.0037 more doctor visits.

### 13.7.16 Marginal treatment effects

The treatment-effects literature measures the effect on an outcome variable of a discrete change in a treatment variable, often a binary variable, in a setting where responses to the treatment variable differ across individuals. Key

measures of treatment effects covered in this book include average treatment effects, average treatment effect on the treated, and local average treatment effect. As [Heckman and Vytlacil \(2007\)](#) have shown, many popular measures of treatment effects can be interpreted as weighted sums of MTE usually estimated in a regression framework. In that sense, MTE is a building block for constructing alternative measures of treatment effects. Chapters 24 and 25 provide several examples.

## 13.8 Model diagnostics

As for the linear model, the modeling process follows a cycle of estimation, diagnostic checks, and model respecification. Here we briefly summarize diagnostics checks with model-specific checks deferred to the models chapters.

### 13.8.1 Goodness-of-fit measures

The  $R^2$  in the linear model does not extend easily to nonlinear models. When fitting by NLS a nonlinear model with additive errors,  $y = m(\mathbf{x}'\boldsymbol{\beta}) + u$ , the residual sum of squares (RSS) plus the model sum of squares (MSS) do not sum to the total sum of squares (TSS). So the three measures MSS/TSS,  $1 - \text{RSS}/\text{TSS}$ , and  $\hat{\rho}_{y,\hat{y}}^2$  [the squared correlation between  $y$  and  $m(\mathbf{x}'\hat{\boldsymbol{\beta}})$ ] differ. By contrast, they all coincide for OLS estimation of the linear model with an intercept. Furthermore, many nonlinear models are based on the distribution of  $y$  and do not have a natural interpretation as a model with an additive error.

A fairly universal measure of association in nonlinear models is  $\hat{\rho}_{y,\hat{y}}^2$ , the squared correlation between  $y$  and  $\hat{y}$ . This has a tangible interpretation and can be used provided that the model yields a fitted value  $\hat{y}$ . This is the case for most commonly used models except multinomial models.

For ML estimators, Stata reports a pseudo- $R^2$  defined as

$$\tilde{R}^2 = 1 - \ln L_{\text{fit}} / \ln L_0 \quad (13.13)$$

where  $\ln L_0$  is the log likelihood of an intercept-only model and  $\ln L_{\text{fit}}$  is the likelihood of the fitted model.

For doctor visits, we have

```

. * Compute pseudo-R-squared after Poisson regression
. qui poisson docvis private chronic female income, vce(robust)
. display "Pseudo-R^2 = " 1 - e(l1)/e(l1_0)
Pseudo-R^2 = .19303857

```

This equals the statistic `Pseudo R2` that is provided as part of `poisson` output; see section [13.3.2](#).

For discrete dependent variables,  $\tilde{R}^2$  has the desirable properties that  $\tilde{R}^2 \geq 0$ , provided that an intercept is included in the model and  $\tilde{R}^2$  increases as regressors are added for models fit by ML. For binary and multinomial models, the upper bound for  $\tilde{R}^2$  is 1, whereas for other discrete data models such as Poisson, the upper bound for  $\tilde{R}^2$  is less than 1. For continuous data, these desirable properties disappear, and it is possible that  $\tilde{R}^2 > 1$  or  $\tilde{R}^2 < 0$  and that  $\tilde{R}^2$  does not increase as regressors are added.

To understand the properties of  $\tilde{R}^2$ , let  $\ln L_{\max}$  denote the largest possible value of  $\ln L(\theta)$ . Then we can compare the actual gain in the objective function due to inclusion of regressors compared with the maximum possible gain, giving the relative gain measure

$$R_{RG}^2 = \frac{\ln L_{\text{fit}} - \ln L_0}{\ln L_{\max} - \ln L_0} = 1 - \frac{\ln L_{\max} - \ln L_{\text{fit}}}{\ln L_{\max} - \ln L_0}$$

In general,  $\ln L_{\max}$  is not known, however, making it difficult to implement this measure (see [Cameron and Windmeijer \[1997\]](#)). For binary and multinomial models, it can be shown that  $\ln L_{\max} = 0$  because perfect ability to model the multinomial outcome gives a probability mass function with a value of 1 and a natural logarithm of 0. Then  $R_{RG}^2$  simplifies to  $\tilde{R}^2$ , given in [\(13.13\)](#). For other discrete models, such as Poisson,  $\ln L_{\max} < 0$  because the probability mass function takes a maximum value of less than 1, so the maximum  $\tilde{R}^2 < 1$ . For continuous density, the log density can exceed 0, so it is possible that  $\ln L_{\text{fit}} > \ln L_0 > 0$  and  $\tilde{R}^2 < 0$ . An example is given as an end-of-chapter exercise.

### 13.8.2 Information criteria for model comparison

For ML models that are nested in each other, we can discriminate between models on the basis of a likelihood-ratio (LR) test of the restrictions that reduce one model to the other; see section [11.4](#).

For ML models that are nonnested, a standard procedure is to use information criteria. Two standard measures are Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (BIC). Different references use different scalings of these measures. Stata uses

$$\begin{aligned} \text{AIC} &= -2 \ln L + 2K \\ \text{BIC} &= -2 \ln L + K \ln N \end{aligned}$$

Smaller AIC and BIC are preferred because higher log likelihood is preferred. The quantities  $2K$  and  $K \ln N$  are penalties for model size.

If the models are actually nested, then an LR test statistic [equals  $\Delta(2 \ln L)$ ] could be used. Then the larger model is favored at a level of 0.05 if  $\Delta(2 \ln L)$  increases by  $\chi^2_{0.05}(\Delta K)$ . By comparison, the AIC favors the larger model if  $\Delta(2 \ln L)$  increases by  $2\Delta K$ , which is a smaller amount [for example, if  $\Delta K = 1$ , then  $2 < \chi^2_{0.05}(1) = 3.84$ ]. The AIC penalty is too small. The BIC gives a larger model-size penalty and is generally better, especially if smaller models are desired.

These quantities are easily displayed using the `estat ic` command. For the Poisson regression with the five regressors, including the intercept, we have

```
. * Report information criteria
. estat ic
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
.	4,412	-22929.9	-18503.55	5	37017.1	37049.06

Note: BIC uses N = number of observations. See [R] BIC note.

The information criteria and LR test require correct specification of the density, so they should instead be used after the `nbreg` command for

negative binomial estimation because the Poisson density is inappropriate for these data.

It is possible to test one nonnested likelihood-based model against another, using the LR test of [Vuong \(1989\)](#). A general discussion of Vuong's test is given in [Greene \(2018, 751\)](#) and [Cameron and Trivedi \(2005, 280–283\)](#). Subsequent research by [Shi \(2015\)](#) shows that the Vuong test can have size distortion and proposes a better test. The journal website includes Stata code to implement this better test.

### 13.8.3 Residuals

Analysis of residuals can be a useful diagnostic tool, as demonstrated in sections [3.6](#) and [6.3](#) for the linear model.

In nonlinear models, several different residuals can be computed. The use of residuals and methods for computing residuals vary with the model and estimator. A natural starting point is the raw residual,  $y_i - \hat{y}_i$ . In nonlinear models, this is likely to be heteroskedastic, leading to use of the Pearson residual  $(y_i - \hat{y}_i)/\hat{\sigma}_i$ , where  $\hat{\sigma}_i^2$  is an estimate of  $\text{Var}(y_i|\mathbf{x}_i)$ . The Pearson statistic is the sum of squared Pearson residuals.

For GLMS defined, in section [13.3.7](#), it is common to use the deviance residual  $d_i = \text{sign}(y_i - \hat{\mu}_i)\sqrt{2\{l(y_i) - l(\hat{\mu}_i)\}}$ , where  $l(y)$  is the log density evaluated at  $\mu = y$  and  $l(\hat{\mu})$  is the log density evaluated at  $\mu = \hat{\mu}$ . For homoskedastic normal errors, this reduces to the raw residual. The deviance statistic is the sum of the squared deviance residuals and is the GLM generalization of the sum of squared residuals.

For the Poisson model fit with the `glm` command, various options of the `predict` postestimation command yield various residuals, and the `mu` option gives the predicted conditional mean. We have

```

. * Various residuals after command glm
. qui glm docvis private chronic female income, family(poisson) vce(robust)
. predict mu, mu
. generate uraw = docvis - mu
. predict upearson, pearson
. predict udeviance, deviance
. predict uanscombe, anscombe
. summarize uraw upearson udeviance uanscombe

```

Variable	Obs	Mean	Std. dev.	Min	Max
uraw	4,412	-3.34e-08	7.408631	-13.31806	125.3078
upearson	4,412	-.0102445	3.598716	-3.519644	91.16232
udeviance	4,412	-.5619514	2.462038	-4.742847	24.78259
uanscombe	4,412	-.5995917	2.545318	-5.03055	28.39791

The Pearson residual has a standard deviation much greater than the expected value of 1 because it uses  $\hat{\sigma}_i^2 = \hat{\mu}_i$ , when in fact there is overdispersion and  $\sigma_i^2$  is several times this. The Anscombe residual uses the transformation of  $y$  closest to normality and then standardizes to mean 0 and variance 1. The deviance and Anscombe residuals are quite similar. The various residuals differ mainly in their scaling; for these data, the pairwise correlations between the residuals exceed 0.92.

Other options of `predict` after `glm` allow for adjustment of deviance residuals and the standardizing and studentizing of the various residuals. The `cooks` and `hat` options aid in finding outlying and influential observations, as for the linear model. For definitions of all of these quantities, see [R] **glm postestimation** or a reference book on GLMS.

Another class of models where residuals are often used as a diagnostic are survival data models; these specialized residuals are presented in section 21.4.4.

### 13.8.4 Model-specification tests

Most estimation commands include a test of overall significance in the header output above the table of estimated coefficients. This is a test of joint significance of all the regressors. Some estimation commands provide

further tests in the output. For example, the `mixed` command includes an LR test against the linear regression model; see sections [6.7](#) and [22.6](#).

More tests may be requested as postestimation commands. Some commands such as `linktest`, to test model specification, are available after most commands. More model-specific commands begin with `estat`. For example, the `poisson` postestimation command `estat gof` provides a goodness-of-fit test.

Discussion of model-specification tests is given in chapter [11](#) and in the model-specific chapters.

## 13.9 Clustered data

In many microeconomics applications, observations can be grouped into clusters, with observations in the same cluster correlated, while observations in different clusters are uncorrelated. For linear models, the various estimation methods and methods for computing standard errors when there is within-cluster correlation were presented in detail in sections [6.4–6.7](#). We now briefly present extension of these methods to nonlinear models, with emphasis on methods for parametric models such as logit, probit and Poisson. These methods are presented further in many of the subsequent models chapters, and similar methods are presented in greater detail for panel data where clustering is on the individual; see chapter 22.

The Poisson model is used as illustration. The analysis here generally extends to other nonlinear models, with two notable exceptions. First, for the Poisson, the population-averaged and random-effects (RE) estimators have the same probability limits. Second, for the Poisson, it is possible to obtain consistent estimates of the fixed-effects (FE) model when there are few observations per cluster.

### 13.9.1 Clustered dataset

We use the same Vietnamese dataset and variables as used in chapter [6](#); see section [6.4.1](#). The dependent variable, the number of direct pharmacy visits (`pharvis`), is a count variable, so Poisson regression is most appropriate. The regressors are the logarithm of household medical expenditures (`lnhhexp`) and the number of illnesses (`illness`). Clustering is on the `commune` variable that identifies the 194 separate villages.

```

. * Read in Vietnam clustered data and delete one household in two communes
. qui use mus206vlss, clear
. drop if lnhhexp > 2.579681 & lnhhexp < 2.579683
(12 observations deleted)
. drop if missing(lnhhexp)
(1 observation deleted)
. summarize pharvis lnhhexp illness commune

```

Variable	Obs	Mean	Std. dev.	Min	Max
pharvis	27,753	.5110439	1.312606	0	30
lnhhexp	27,753	2.60262	.6245493	.0467014	5.405502
illness	27,753	.6218427	.8995018	0	9
commune	27,753	101.514	56.27264	1	194

### 13.9.2 Pooled or population-averaged models

For individual  $i$  in cluster  $g$  (of  $G$  clusters), a parametric model for the conditional density of  $y_{ig}$  given regressors is typically of the single-index form

$$f(y_{ig} | \mathbf{x}_{ig}) = f(\alpha + \mathbf{x}'_{ig}\beta, \gamma); \quad i = 1, \dots, N_g, \quad g = 1, \dots, G$$

where we have separated out the intercept from other regressors and  $\gamma$  denotes additional parameters such as variance parameters.

The pooled or population-averaged approach continues to assume that this conditional density for a single observation is correct, despite within-cluster correlation, but corrects standard errors for within-cluster correlation.

#### Pooled Poisson

The pooled Poisson estimator is the usual Poisson estimator, with cluster-robust standard errors used rather than default or heteroskedastic-robust standard errors. We have

```

. * Poisson estimation with cluster--robust standard errors clustered on commune
. poisson pharvis lnhhexp illness, nolog vce(cluster commune)
Poisson regression                                         Number of obs = 27,753
                                                               Wald chi2(2) = 591.84
                                                               Prob > chi2 = 0.0000
Log pseudolikelihood = -26217.797                         Pseudo R2 = 0.1769
                                                               (Std. err. adjusted for 194 clusters in commune)

```

pharvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lnhhexp	.0175333	.0472699	0.37	0.711	-.075114 .1101806
illness	.6806127	.0286861	23.73	0.000	.624389 .7368365
_cons	-1.412108	.1451221	-9.73	0.000	-1.696542 -1.127674

Controlling for within-commune correlation leads to larger standard errors, smaller  $t$  statistics, and larger  $p$ -values. From output not given, the default standard errors for variables `lnhhexp` and `illness` are, respectively, 0.0135 and 0.0053, whereas the heteroskedastic–robust standard errors are 0.0268 and 0.0264.

### Generalized estimating equations

As detailed in section [13.3.1](#), consistency of the Poisson quasi-MLE requires only that the conditional mean be correctly specified; the distribution need not be the Poisson. This robustness property is shared by a range of commonly used models, including logit and probit, that the statistics literature calls GLMS.

For clustered data, the GLM framework is extended to the generalized estimating equations framework. This provides feasible generalized least-squares estimation of nonlinear models given specification of a model for the within-cluster correlation to obtain potentially more efficient parameter estimates. One can again obtain cluster–robust standard-error estimates that guard against misspecification of the model for within-cluster correlation, provided there are many clusters.

A natural model for within-cluster correlation is a model of equicorrelation with  $\text{Cor}(y_{ig}, y_{jg}) = \rho$  for individuals  $i$  and  $j$  in the same cluster. This assumption is also called one of exchangeability.

The model can be fit using the `xtgee` command detailed in section 22.4.4. We have

```
. * Generalized estimating equation estimation with cluster--robust st. errors
> clustered on commune
. xtset commune
Panel variable: commune (unbalanced)
. xtgee pharvis lnhhexp illness, nolog family(poisson)
> corr(exchangeable) vce(robust)

GEE population-averaged model
Group variable: commune
Family: Poisson
Link: Log
Correlation: exchangeable
Number of obs      = 27,753
Number of groups  =    194
Obs per group:
min =          51
avg =       143.1
max =       206
Wald chi2(2)      = 515.86
Prob > chi2        = 0.0000
Scale parameter = 1
(Std. err. adjusted for clustering on commune)
```

pharvis	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
lnhhexp	-.0305989	.0078615	-3.89	0.000	-.0460072	-.0151906
illness	.2697244	.011886	22.69	0.000	.2464282	.2930207
_cons	.9510729	.0793079	11.99	0.000	.7956323	1.106513

The parameter estimates in this example with a very simple model are much more precisely fit and considerably different from those obtained using the `poisson` command. This difference is due to the very high estimate of within-cluster correlation: the postestimation command `estat wcorrelation` yields  $\hat{\rho} = 0.879$ . The use of the `vce(robust)` option following an `xt` command leads to cluster-robust standard errors being reported. In this example, there is a substantial efficiency gain.

### 13.9.3 RE and mixed models

An RE model allows the intercept to vary across cluster, so the conditional density for observations in the  $g$ th cluster becomes

$$f(y_{ig} | \mathbf{x}_{ig}, \alpha_g) = f(\alpha_g + \mathbf{x}'_{ig}\beta, \gamma); \quad i = 1, \dots, N_g, \quad g = 1, \dots, G \quad (13.14)$$

The standard assumption is that  $\alpha_g$  is normally distributed with mean 0 and variance  $\sigma_\alpha^2$ .

It is assumed that observations within cluster are independent given addition of the cluster-specific intercept  $\alpha_g$ . It follows that the joint density for all observations in cluster  $g$  is the product of the individual densities. Integrating out  $\alpha_g$  yields

$$\begin{aligned} & f(y_{1g}, \dots, y_{N_g g} | \mathbf{x}_{1g}, \dots, \mathbf{x}_{N_g g}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma_\alpha^2) d\alpha_g \\ &= \int \left\{ \prod_{i=1}^{N_g} f(y_{ig} | \mathbf{x}_{ig}, \alpha_g, \boldsymbol{\beta}, \boldsymbol{\gamma}) \right\} \phi(\alpha_g | \sigma_\alpha^2) d\alpha_g \end{aligned} \quad (13.15)$$

where  $\phi(\alpha_g | \sigma_\alpha^2)$  is the  $N(0, \sigma_\alpha^2)$  density. This one-dimensional integral is computed using Gauss–Hermite quadrature.

For the Poisson RE model,  $y_{ig}$  has Poisson distribution with mean  $\exp(\mathbf{x}'_{ig} \boldsymbol{\beta} + \alpha_g)$ , where  $\alpha_g$  is  $N(0, \sigma_\alpha^2)$  distributed. We use the `xtpoisson` command, a command with syntax similar to the `xtreg` command, with the `re` and `normal` options to obtain

```

. * RE estimation with cluster--robust st. errors clustered on commune
. xtpoisson pharvis lnhhexp illness, re normal nolog vce(robust)
Calculating robust standard errors ...
Random-effects Poisson regression                               Number of obs      = 27,753
Group variable: commune                                     Number of groups =     194
Random effects u_i ~ Gaussian                                Obs per group:
                                                               min =        51
                                                               avg =    143.1
                                                               max =    206
Integration method: mvaghermite                            Integration pts. =      12
                                                               Wald chi2(2)     = 1141.31
Log pseudolikelihood = -24324.99                           Prob > chi2      = 0.0000
                                                               (Std. err. adjusted for 194 clusters in commune)

```

pharvis	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
lnhhexp	-.1422882	.0404454	-3.52	0.000	-.2215596	-.0630168
illness	.7151174	.0213484	33.50	0.000	.6732753	.7569596
_cons	-1.234728	.1177822	-10.48	0.000	-1.465577	-1.003879
/lnsig2u	-.8734784	.1612943			-1.189609	-.5573474
sigma_u	.6461399	.0521093			.5516703	.7567868

Compared with results obtained using the `poisson` command, the coefficient estimates have smaller standard errors, and the coefficient of `lnhhexp` is negative and statistically significant.

The same results, aside from small numerical differences, are obtained using the `mepoisson` command, with only the intercept varying.

```

. * Mixed estimation with cluster--robust st. errors clustered on commune
. mepoisson pharvis lnhexp illness || commune: , nolog vce(robust)

Mixed-effects Poisson regression                               Number of obs      =    27,753
Group variable: commune                                     Number of groups   =       194
                                                               Obs per group:
                                                               min =            51
                                                               avg =          143.1
                                                               max =          206
Integration method: mvaghermite                           Integration pts. =        7
                                                               Wald chi2(2)     =    1141.34
Log pseudolikelihood = -24324.99                         Prob > chi2      =    0.0000
                                                               (Std. err. adjusted for 194 clusters in commune)

```

pharvis	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
lnhexp	-.142289	.0404443	-3.52	0.000	-.2215583	-.0630197
illness	.7151175	.0213481	33.50	0.000	.6732759	.756959
_cons	-1.234721	.1177576	-10.49	0.000	-1.465522	-1.003921
commune						
var(_cons)	.4175234	.0673298			.30438	.5727242

Mixed-effects model commands such as `mepoisson` additionally allow slope coefficients to vary across clusters; see section 23.4.

RE models that allow for endogeneity or sample selection, or both, can be fit using the extended regression model commands `xteregress`, `xteprobit`, `xteoprobit`, and `xtintreg`; see section 23.7.

In the special case of the Poisson, an alternative RE model specifies  $y_{ig}$  to be Poisson distributed with mean  $\exp(\mathbf{x}'_{ig}\beta + \alpha_g)$ , where  $\exp(\alpha_g)$  is gamma distributed. In that case, there is a closed-form solution to the integral ([13.15](#)). This estimator, presented in section 22.6.5, is obtained using the `re` option of the `xtpoisson` command.

### 13.9.4 FE model

The FE model treats  $\alpha_g$  in ([13.14](#)) as a cluster-specific fixed effect that, unlike in the RE model, may be correlated with regressors. This then introduces  $G$  additional parameters  $\alpha_1, \dots, \alpha_G$ .

In general, consistent estimation of the parameters  $\beta$  requires consistent estimation of each parameter  $\alpha_g$ , which in turn requires  $N_g \rightarrow \infty$ . So in general, consistent estimation of the FE model requires many observations per cluster, a limitation known as the incidental parameters problem. There are three notable exceptions to this limitation, namely, the linear regression model, the Poisson model, and the logit model. For those models, one can consistently estimate the parameters  $\beta$  in an FE model even if there are few observations per cluster.

Even when the parameters  $\beta$  can be consistently estimated, consistent estimates of MES and predictions cannot be obtained if the fixed effects  $\alpha_g$  enter nonlinearly and are not consistently estimated. Thus, if  $E(y_{ig}|\mathbf{x}_{ig}, \alpha_g) = g(\alpha_g + \mathbf{x}'_{ig}\beta)$ , for example, we need a consistent estimate of  $\alpha_g$ . At the same time, if regressors enter in this index form, then we at least know the relative effects of regressors, so if  $\beta_j > \beta_k$ , then the ME of an increase in the  $j$ th regressor is twice that of the  $k$ th regressor. And the sign of the effect can be determined if  $g(\cdot)$  is monotonic.

For the Poisson FE model, we use the `xtpoisson` command with the `fe` option to obtain

```
. * FE estimation with cluster--robust st. errors clustering on commune
. xtpoisson pharvis lnhhexp illness, fe nolog vce(robust)
note: 1 group (94 obs) dropped because of all zero outcomes

Conditional fixed-effects Poisson regression           Number of obs      = 27,659
Group variable: commune                            Number of groups   =     193
                                                       Obs per group:
                                                       min =          51
                                                       avg =       143.3
                                                       max =        206
                                                       Wald chi2(2)    = 1125.50
Log pseudolikelihood = -23340.71                  Prob > chi2      = 0.0000
                                                       (Std. err. adjusted for clustering on commune)
```

	Robust					
pharvis	Coefficient	std. err.	z	P> z	[95% conf. interval]	
lnhhexp	-.1546582	.0411747	-3.76	0.000	-.2353591	-.0739573
illness	.716148	.0215378	33.25	0.000	.6739347	.7583614

In this example, the results are fairly similar to those for the Poisson RE model.

An alternative brute-force estimation procedure includes indicator or dummy variables for each cluster as regressor. In the current example, this leads to the inclusion of 192 additional regressors, so to suppress output, we use the `quietly` prefix in estimation and `estimates store` and commands. We obtain

```
. * Dummy variables estimation with robust st. errors clustered on commune
. qui poisson pharvis lnhhexp illness i.commune, vce(cluster commune)
. estimates store POISSONDV
. estimates table POISSONDV, keep(lnhhexp illness) b(%10.6f) se stats(N)
```

Variable	POISSONDV
lnhhexp	-0.154657 0.041281
illness	0.716149 0.021594
N	27753

Legend: b/se

We obtain the same results, aside from rounding error in computation, as those for the `xtpoisson, fe` command.

Note that while this latter method can always be implemented, subject to computational limits if there are too many clusters, for all estimators aside from OLS and Poisson, consistent estimation requires many observations, say, at least 30, per cluster. Recent research has proposed methods that correct for this bias, or more precisely, the inconsistency, that arises when there are few observations per cluster. The community-contributed commands `xtprobitfe` and `xtlogitfe` ([Cruz-Gonzalez, Fernández-Val, and Weidner 2017](#)) do so for probit and logit models; see section 22.4.8.

### 13.9.5 Correlated RE model

An alternative approach when fixed effects are present is to use the correlated RE model, introduced for linear panel models in section [8.7.4](#) and presented for nonlinear panel models in section 22.2.4. Then

$\alpha_g = \bar{\mathbf{x}}'_{1g} \gamma + \eta_g$ , where  $\eta_g$  is a random effect and  $\mathbf{x}_{1g}$  denotes just those regressors that vary within cluster.

For the current example, using the `xtpoisson, re` command, we obtain

```

. * Correlated RE estimation with cluster-specific effects
. bysort commune: egen avelnhhexp = mean(lnhhexp)
. by commune: egen aveillness = mean(illness)
. xtpoisson pharvis lnhhexp illness avelnhhexp aveillness, re nolog vce(robust)

Random-effects Poisson regression                         Number of obs     =  27,753
Group variable: commune                                Number of groups  =      194
Random effects u_i ~ Gamma                           Obs per group:
                                                       min =          51
                                                       avg =      143.1
                                                       max =      206
Log pseudolikelihood = -24304.828                      Wald chi2(4)     = 1691.40
                                                               Prob > chi2    = 0.0000
                                                               (Std. err. adjusted for clustering on commune)

```

pharvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
lnhhexp	-.1550693	.041469	-3.74	0.000	-.2363472	-.0737915
illness	.7139736	.0394978	18.08	0.000	.6365593	.791388
avelnhexp	.4284912	.1016099	4.22	0.000	.2293395	.6276428
aveillness	.3847358	.1810005	2.13	0.034	.0299814	.7394902
_cons	-2.392214	.3382878	-7.07	0.000	-3.055246	-1.729182
/lnalpha	-1.12778	15.67529			-31.85078	29.59522
alpha	.3237511	5.074892			1.47e-14	7.13e+12

LR test of alpha=0: chibar2(01) = 3494.51 Prob >= chibar2 = 0.000

The coefficients of  $-0.1551$  and  $0.7140$  are very close to the Poisson FE estimates of  $-0.1547$  and  $0.7161$ , though, unlike the linear case, they are not identical. Using normally distributed random effects (the `re ml` option of `xtpoisson`) yields very similar results.

Some practitioners advocate this as a simple way to control for cluster-specific fixed effects in models for which, unlike the linear and Poisson models, FE estimation is not possible when there are few observations per cluster.

## 13.10 Additional resources

A complete listing of estimation commands can be obtained by typing `help` estimation commands. For `poisson`, for example, see the entries **[R] poisson** and **[R] poisson postestimation** and the corresponding online help. Useful Stata commands include `predict`, `predictnl`, `margins`, `lincom`, and `nlcom`.

Graduate econometrics texts give considerable detail on estimation of nonlinear models and less on prediction and computation of MES. See **[R] margins**, and for details on unconditional standard errors of MES, see *Obtaining margins with survey data and representative samples* and *Methods and formulas*.

## 13.11 Exercises

1. Fit the Poisson regression model of section 13.3 by using the `poisson`, `n1`, and `glm` commands. In each case, report default standard errors and robust standard errors. Use the `estimates store` and `estimates table` commands to produce a table with the six sets of output, and discuss.
2. In this exercise, we use the medical expenditure data of section 3.5 with the dependent variable  $y = \text{totexp}/1000$  and regressors the same as those in section 3.5. We suppose that  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ , which ensures that  $E(y|\mathbf{x}) > 0$ . The obvious estimator is NLS, but the Poisson MLE is also consistent if  $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$  and does not require that  $y$  be integer values. Repeat the analysis of question 1 with these data.
3. Use the same medical expenditure data as in exercise 2. Compare the different standard errors obtained with `poisson`'s `vce()` option with the `vcetypes` `oim`, `opg`, `robust`, `cluster` `clustvar`, and `bootstrap`. For clustered standard errors, cluster on `age`. Comment on your results.
4. Consider Poisson regression of `docvis` on an `intercept`, `private`, `chronic`, and `income`. This is the model of section 13.5, except `female` is dropped. Find the following: the sample average prediction of `docvis`; the average prediction if we use the Poisson estimates to predict `docvis` for males only; the prediction for someone who is privately insured, has a chronic condition, and has an income of \$20,000 (so `income=20`); and the sample average of the residuals.
5. Continue with the same data and regression model as in exercise 4. Provide a direct interpretation of the estimated Poisson coefficients. Find the ME of changing regressors on the conditional mean in the following ways: the MEM using calculus methods for continuous regressors and finite-difference methods for discrete regressors; the MEM using calculus methods for all regressors; the AME using calculus methods for continuous regressors and finite-difference methods for discrete regressors; the AME using calculus methods for all regressors; and the MER for someone who is privately insured, has a chronic condition, and has an income of \$20,000 (so `income=20`).
6. Consider the following simulated data example. Generate 100 observations of  $y \sim N(0, 0.001^2)$ , first setting the seed to 10101. Let  $x = \sqrt{y}$ . Regress  $y$  on an intercept and  $x$ . Calculate the pseudo- $R^2$

defined in (13.13). Is this necessarily a good measure when data are continuous?

7. Consider the negative binomial regression of `docvis` on an intercept, `private`, `chronic`, `female`, and `income`, using the `nbreg` command (`replace poisson by nbreg`). Compare this model with one with `income` excluded. Which model do you prefer on the grounds of 1) AIC, 2) BIC, and 3) an LR test using the `lrtest` command?



# **Chapter 14**

## **Flexible regression: Finite mixtures and nonparametric**

## 14.1 Introduction

In this chapter, we consider regression models more flexible than the linear regression model for the conditional mean or the fully parametric linear regression with normally distributed errors.

We begin with flexible fully parametric models for the conditional density of  $y$  given  $\mathbf{x}$  based on finite mixtures of the normal distribution; see [Ahamada and Flachaire \(2010\)](#). This same approach, one that has a long history in statistics, can be applied to other types of data. Finite mixtures for a Poisson model for count data are presented in section 20.5, and many examples of finite mixture models (FMM) are presented in section 23.3.

We then consider flexible methods to model the conditional mean in the linear model. One way to do so is to use polynomials in underlying variables. The most obvious way to do so is to use global polynomials that fit a single polynomial model throughout the range of the regressors. For example, a cubic model specifies  $E(y|x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ . Even more flexibility can be obtained by using higher-order polynomials, but these can be overly responsive to outlying observations.

An alternative approach, widely used in applied statistics, is to flexibly model the conditional mean using regression splines. For example, a cubic spline fits separate cubic polynomials over subranges of  $x$  in such a way that these separate polynomials are connected (or splined) and lead to a continuous function for  $E(y|x)$ .

While we present global polynomials and splines for models with conditional mean of the form  $\mathbf{z}'_i \boldsymbol{\gamma}$ , these methods generalize to other models for the way that regressors enter, such as logit or probit or the exponential conditional mean function  $\exp(\mathbf{z}'_i \boldsymbol{\gamma})$ .

Finally, we introduce nonparametric modeling of the conditional mean in the case of a single regressor. Then no functional form is specified for  $E(y|x)$ . Instead, estimation is by kernel-weighted methods such as local polynomial regression or by series methods such as polynomials or regression splines where the polynomial degree or the number of knots is data determined. The chapter also introduces semiparametric models where the conditional mean is partially parameterized. Nonparametric regression methods for single and multiple regressors, and semiparametric regression methods, are presented in much greater detail in chapter 27.

## 14.2 Models based on finite mixtures

The essential motivation behind a mixture model is that it departs from the assumption of a parametric data-generating process (DGP) based on a single distribution. By contrast, a mixture of distributions is generally more flexible and likely to provide a better approximation to the true unknown distribution. Adding components to the model may improve the statistical fit of the specified model. A second motivation is that a mixture model can capture discrete population heterogeneity by allowing the different components of the mixture to have different moments or moment functions. Finally, when mixture models are fit by maximum likelihood (ML), the estimators have certain optimality properties, and inference can proceed along well-established lines.

### 14.2.1 Univariate case

The main focus here will be on regression mixtures. But to aid intuition, we shall also briefly cover univariate mixtures.

An example of a univariate two-component mixture of normals is  $\pi N(\mu_1, \sigma_1^2) + (1 - \pi)N(\mu_2, \sigma_2^2)$ , which mixes draws from subpopulations  $N(\mu_1, \sigma_1^2)$  and  $N(\mu_2, \sigma_2^2)$  in proportions  $\pi$  and  $1 - \pi$ , respectively. By allowing the location and scale parameters to differ across subpopulations, the mixture model captures discrete population heterogeneity.

Note that the mixture model is a weighted sum of densities; it is not a weighted sum of random variables.

The following example contrasts an FMM, a mixture of  $N(4, 1)$  with probability 0.75 and  $N(8, 1)$  with probability 0.25, with a weighted sum of normally distributed random variables, the random variable  $Z = 0.75X + 0.25Y$ , where  $X \sim N(4, 1)$  and  $Y \sim N(8, 1)$ .

```

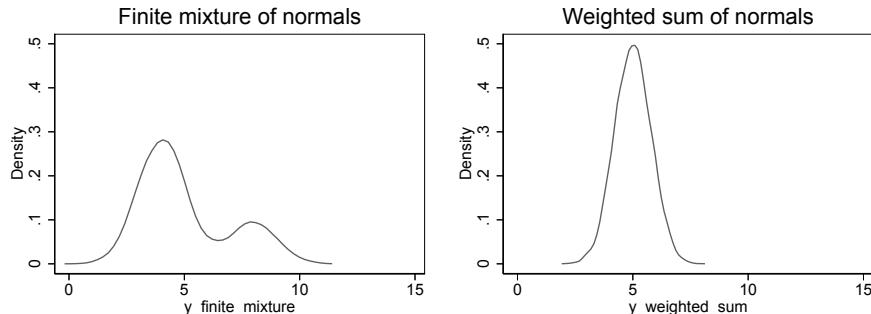
. * Compare mixture of normal densities with r.v. that is weighted sum of normals
. set obs 10000
Number of observations (_N) was 0, now 10,000.
. generate y_finite_mixture = rnormal(4,1)
. replace y_finite_mixture = rnormal(8,1) if runiform()>0.75
(2,516 real changes made)
. kdensity y_finite_mixture, title("Finite mixture of normals") note(" ")
. generate y_weighted_sum = 0.75*rnormal(4,1)+0.25*rnormal(8,1)
. kdensity y_weighted_sum, title("Weighted sum of normals") note(" ")
. sum y_finite_mixture y_weighted_sum

```

Variable	Obs	Mean	Std. dev.	Min	Max
y_finite_m^e	10,000	5.011067	2.008076	.2235037	11.00953
y_weighted^m	10,000	5.004159	.7893892	2.164636	8.140759

From the summary statistics, the two random variables appear to have the same mean but different standard deviations. We leave it as an exercise to derive the theoretical means and variances of the two random variables.

From the first panel of figure 14.1, we see the finite mixture distribution is bimodal, with peaks around 4 and 8. The second panel of the figure shows that the weighted sum of independent normal random variables is normally distributed with mean of around 5.



**Figure 14.1.** Finite mixture compared with weighted sum of normal random variables

Formally, a finite mixture of  $C$  distributions is written as

$$f(y_i|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_C) = \sum_{j=1}^C \pi_j f_j(y_i|\boldsymbol{\theta}_j), \quad \sum_{j=1}^C \pi_j = 1, \quad 0 < \pi_j < 1$$

Typically, the mixture components  $f_j(\cdot)$  of a mixture are in the same parametric family, although formally there is no requirement that they should be. A special case with  $C = 2$  is

$f(y_i|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \pi_1 f_1(y_i|\boldsymbol{\theta}_1) + (1 - \pi_1) f_2(y_i|\boldsymbol{\theta}_2)$ . A mixture model with a finite number of components is called an FMM. Because the mixture components are hidden, not directly observed, the model is also known as the latent class model. The mixture model is often interpreted as a representation of different “types” in the population, each type corresponding to a different component.

In practice, the number of components, denoted by  $C$ , is another unknown parameter. Rather than directly estimating  $C$ , one fits the model conditionally on an assumed value of  $C$ . This leads to the problem of choosing the best value of  $C$  according to some criterion.

We may think of each distribution as specific to one class. The probability weight of each class is a *prior probability*. As the sample size grows, we expect to get more observations from each component distribution, so it becomes easier to identify additional mixture components.

### 14.2.2 Regression case

In the regression case a finite mixture of  $C$  distributions is written as

$$f(y_i|\mathbf{x}_i, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_C) = \sum_{j=1}^C \pi_j f_j(y_i|\mathbf{x}_i, \boldsymbol{\theta}_j), \quad \sum_{j=1}^C \pi_j = 1, \quad 0 < \pi_j < 1$$

where in the simplest case the  $\pi_j$  are scalar parameters to be estimated.

Further flexibility of this specification results from a generalization that treats the  $\pi_j$  as a function of observable variables, say,  $\mathbf{z}$ . To ensure that the

weights lie between zero and one, and sum to one, we use a multinomial logit model. We replace  $\pi_j$  by  $\pi_{ji}$ , where

$$\pi_{ji} = \frac{\exp(\mathbf{z}'_i \boldsymbol{\gamma}_j)}{1 + \exp(\mathbf{z}'_i \boldsymbol{\gamma}_2) + \cdots + \exp(\mathbf{z}'_i \boldsymbol{\gamma}_C)}$$

and we use the normalization  $\boldsymbol{\gamma}_1 = \mathbf{0}$ . The conditional mean for the  $i$ th individual is a weighted sum,

$$E(y_i | \mathbf{x}_i) = \sum_{j=1}^C \pi_{ji} \mu_{ji}$$

where  $\mu_{ji} = E_j(y_i | \mathbf{x}_i)$  is the conditional mean of the  $j$ th group. More generally, the  $k$ th central moment  $E(y_i^k | \mathbf{x}_i)$  is a weighted sum of the  $k$ th central moment of each component density.

For the simpler case where  $\pi_{ji} = \pi_j$  for all  $i$ , the overall marginal effect is a weighted sum of the marginal effects for the  $C$  types:

$$\frac{\partial E(y_i | \mathbf{x}_i)}{\partial \mathbf{x}_i} = \sum_{j=1}^C \pi_j \frac{\partial \mu_j}{\partial \mathbf{x}_i}$$

FMMS are closely related to intrinsic classification models and models with clustering. Endogenous switching models in macroeconomics also have a finite mixture interpretation; see [Frühwirth-Schnatter \(2006\)](#). The mixture model is easier to interpret when the number of components is small and the components are well separated, that is, do not overlap much. However, such ease of interpretation will usually diminish as more components are added to obtain a better fit to the data.

### 14.2.3 Regression example

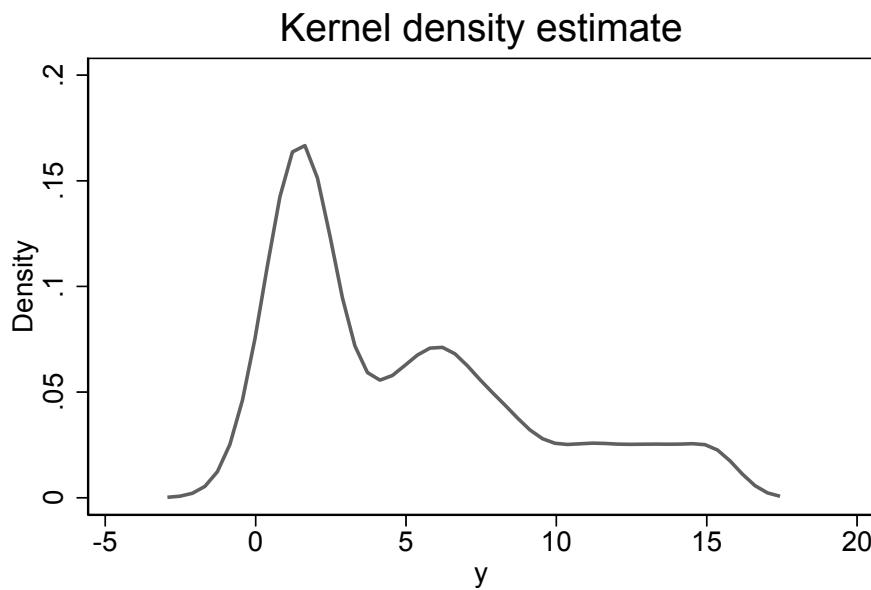
As an example, we generate data for a mixture of three distributions that depend on a single regressor  $x$ . We have

```
. * DGP for regression 3-component mixture of normals: Well-separated components
. clear all
. set obs 10000
Number of observations (_N) was 0, now 10,000.
. set seed 10101
. generate x = runiform()
. generate y = 1 + 1*x + rnormal()
. generate class = runiform()
. replace y = 4 + 4*x + .8*rnormal() if class > 0.5
(4,956 real changes made)
. replace y = 8 + 8*x + .4*rnormal() if class > 0.8
(2,049 real changes made)
```

This leads to the following summary statistics and density.

```
. * Regression mixture of normals: Summary statistics and density
. summarize y x
Variable | Obs Mean Std. dev. Min Max
y | 10,000 4.965995 4.359061 -2.31192 16.81672
x | 10,000 .4997462 .288546 .0000276 .9999758
. kdensity y, kernel(gaussian) lwidth(medthick) note(" ") scale(1.2)
```

From figure 14.2, the three components are still observable, despite the presence of the regressor  $x$ . More generally, however, the components are not as well separated in the presence of regressors.



**Figure 14.2.** Finite mixture density of three normal distributions

If one fits a one-component normal regression by ordinary least-squares (OLS) to these data generated by a mixture of normals, the estimated conditional mean is a weighted sum of the component means; that is,  $\mathbf{x}'\boldsymbol{\beta} = \sum_{j=1}^3 \pi_j \mathbf{x}'\boldsymbol{\beta}_j$ . Hence, the one-component model identifies  $\boldsymbol{\beta}$ , which may not be the target parameter. The fitted one-component model is interpreted as the closest one-component distribution (in the Kullback–Leibler metric) that minimizes the expected distance from the underlying mixture model.

#### 14.2.4 Modeling considerations

The mixture representation is invariant with respect to the labeling of the components. But a rule for labeling is nevertheless beneficial because it enables comparison of the components from specifications with different values of  $C$ . The criterion we will use will be the conditional mean function evaluated at each set of component-specific parameter values.

As a part of a descriptive overview, it is standard to begin by plotting a histogram or a kernel density of the outcome of interest. A multimodal shape may provide a hint of the presence of mixture. However, such an eyeball

test, while potentially useful when considering univariate mixtures, is not conclusive in the context of a regression model where the mixture hypothesis pertains not to  $y$  per se but to  $y|x$ , that is, residuals from the regression of  $y$  on  $x$ . Potentially, multimodality of the distribution of residuals can be investigated. However, a failure to detect multimodality does not necessarily invalidate the mixture hypothesis if the mixture components are not well separated.

The typical way for a practitioner to proceed is to 1) select the mixing distribution; 2) fix an initial value of  $C$ ; 3) fit the model using ML; 4) increase  $C$ ; 5) fit the new model; and 6) use a model-selection criterion to select the preferred model.

#### 14.2.5 The `fmm` prefix

The Stata `fmm` prefix, whose earlier community-contributed version was due to [Deb \(2007\)](#), supports ML estimation of finite mixtures of both continuous and discrete distributions. The current version includes more than 15 families of distributions. Further applications of this command are given in sections 23.2 and 23.3.

In this chapter, we consider only mixtures of normal regression models. In that case, the applicable model command is `regress`, and we give command

```
fmm # [if] [in] [weight] [, fmmopts]: regress depvar [indepvars]
    [, options]
```

where `#` refers to the number of normal components in the specification.

For example, for a two-component normal regression model with dependent variable  $y$ , model regressors in global macro `xlist`, and constant class probability  $\pi$ , we give command

```
. * Example of command fmm for two-component normal model
. fmm 2: regress y $xlist
```

Specifying `lcprob()` in `fmmopts` allows the latent class probabilities  $\pi_j$  to be parameterized as a function of observable variables. For example, if

these are in global macro `lcprobzlist`, we give command

```
. * Example of command fmm for two-component normal with varying class  
> probabilities  
. fmm 2, lcprob($lcprobzlist): regress y $xlist
```

Additional flexibility in specification can be obtained by allowing the specification of the latent class or regression function, or both, to depend on different sets of regressors for different components of the model. This family of models is also known as a “mixture-of-experts” model and has a well-established place in machine learning and artificial intelligence literature; see [Jacobs et al. \(1991\)](#) and [Jordan and Jacobs \(1994\)](#).

Interpretation of estimates of FMMS can be difficult. Useful postestimation commands include `predict`, `estat lcmean`, `margins`, and `estat lcprob`.

### 14.2.6 Computational considerations

The expectation–maximization algorithm (see [Cameron and Trivedi \[2005\]](#), 345–347]) is used to find starting values. Then, gradient-based optimization methods such as Gauss–Newton are used; see chapter 16 on numerical optimization.

The `startvalues()` option provides several methods for specifying starting values. The default assigns each observation to an initial latent class that is determined by running a factor analysis on the specified regressors. Alternatively, start values can be specified using the `from()` option.

Gradient-based algorithms work well when the objective function is well approximated by a quadratic function in the parameters. For mixture models, convergence cannot be guaranteed, because the likelihood function is not locally concave, and a warning message is given when this condition arises.

A failure to fit a specified mixture model may occur for several reasons. First, the log likelihood may have several local maximums such that the starting values play an important role. Second, the mixture components may not be well separated, which would make reliable identification of a component difficult. Third, the specified mixture model may have one or more components with small associated mixture weights  $\pi_j$ ; this would imply that the model is overparameterized and the parameters of at least one of the components cannot be identified. The possibility of overparameterization is greater when the model has more components and also when the mixture-of-experts specification is used.

When the main motive behind the mixture model is to improve an approximation to the underlying distribution, a model with a small number of components, such as 2 or 3, has the attraction of being a parsimonious specification. However, when the underlying motivation is to finely define the structure in terms of “latent types,” a more highly parameterized model may be preferred. But the empirical success of such an exercise depends upon the available sample size and the extent of separation between the types.

## 14.3 FMM example: Earnings of doctors

We illustrate empirical application of the `fmm` prefix using data on the natural logarithm of annual earnings of doctors derived from three waves of the Australian longitudinal survey, Medicine in Australia: Balancing Employment and Life.

### 14.3.1 Data and log-linear regression

The regression model takes log of earnings (in Australian dollars) as the dependent variable (`logyearn`). The regressors are log of annual hours worked (`logyhrs`), a gender indicator (`female`), the presence of a child under 5 years of age (`childu5`), years of work experience (`expr`), square of `expr` (`exprsq`), the number of other postgraduate qualifications (`pgradoth`), and the size of the medical practice where the doctor works (`pracsize`).

```
. * Log earnings of doctors example: Read in data and give summary statistics  
. qui use mus214mabelfmm, clear  
. global xlist logyhrs female childu5 expr exprsq pgradoth pracsize  
. summarize logyearn $xlist, sep(0)
```

Variable	Obs	Mean	Std. dev.	Min	Max
logyearn	5,384	11.92434	.6931311	7.824046	13.81751
logyhrs	5,384	7.514087	.4925595	3.951244	8.556414
female	5,384	.4686107	.4990601	0	1
childu5	5,384	.1586181	.3653535	0	1
expr	5,384	23.26337	10.58021	0	58
exprsq	5,384	653.1045	535.3212	0	3364
pgradoth	5,384	.5815379	.788196	0	4
pracsize	5,384	3.317979	1.189747	1	5
. tabstat logyearn, stat(mean p50 sd skewness kurtosis)					
Variable	Mean	p50	SD	Skewness	Kurtosis
logyearn	11.92434	11.95761	.6931311	-.5159499	4.030809

The detailed statistics for `logyearn` indicate that the normal distribution is a reasonable starting point for these data because the distribution is reasonably symmetric and the kurtosis statistic of 4.03 is not too far from 3.0. A kernel

density estimate, not given, is close to the normal, albeit with a longer left tail.

A standard log-linear regression provides a benchmark for comparison with FMMs with more than one component.

```
. * Standard OLS log-earnings regression (= 1 component FMM) with robust
> standard errors
. regress logyearn $xlist, vce(robust)

Linear regression
Number of obs      =      5,384
F(7, 5376)        =     431.63
Prob > F          =     0.0000
R-squared          =     0.4456
Root MSE           =     .51643
```

logyearn	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
logyhrs	.8035603	.0267656	30.02	0.000	.7510888	.8560318
female	-.2607014	.0187799	-13.88	0.000	-.2975177	-.2238852
childu5	.1174038	.0199764	5.88	0.000	.0782419	.1565656
expr	.01334	.0026442	5.04	0.000	.0081563	.0185237
exprsq	-.0002714	.0000541	-5.01	0.000	-.0003775	-.0001653
pgradoth	.0291406	.0088829	3.28	0.001	.0117265	.0465547
pracsiz	.0248134	.0066149	3.75	0.000	.0118455	.0377812
_cons	5.75751	.2124388	27.10	0.000	5.341044	6.173976

```
. estimates store fmm1
```

The fit is quite good for cross-sectional earnings data, with  $R^2 = 0.45$ . The regressors are all statistically significant at level 0.05 and have the expected signs. Earnings increase with experience up to a turning point at 24.7 years of experience.

A graph of the kernel density of residuals from this regression (not reported here) is almost symmetric but with fatter tails. In contrast to the generated data example given previously, this residual density plot gives no hint of multimodality and suggests weak separation of mixture components in any mixture model.

### 14.3.2 Two-component model estimates

We next fit the corresponding two-component FMM.

```
. * Two-component mixture of (log) normals: ML estimates
. fmm 2, nolog vce(robust): regress logyearn $xlist
```

Finite mixture model Number of obs = 5,384  
Log pseudolikelihood = -3820.4547

	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
1.Class	(base outcome)				
2.Class					
_cons	1.145835	.4280244	2.68	0.007	.3069222 1.984747

Class: 1  
Response: logyearn  
Model: regress

	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
logyearn					
logyhrs	.3654256	.0952684	3.84	0.000	.178703 .5521482
female	-.3316877	.0725441	-4.57	0.000	-.4738714 -.1895039
childu5	.0892002	.0620325	1.44	0.150	-.0323812 .2107816
expr	.0218933	.0088244	2.48	0.013	.0045979 .0391888
exprsq	-.0006673	.0001756	-3.80	0.000	-.0010114 -.0003231
pgradoth	.104355	.0338084	3.09	0.002	.0380917 .1706184
pracsize	.1144076	.0282952	4.04	0.000	.0589499 .1698652
_cons	8.630774	.7334559	11.77	0.000	7.193227 10.06832
var(e.logyearn)	.4314492	.1020731			.2713631 .6859754

Class: 2  
Response: logyearn  
Model: regress

	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
logyearn					
logyhrs	1.018329	.0611357	16.66	0.000	.8985049 1.138152
female	-.2171311	.0254682	-8.53	0.000	-.2670478 -.1672144
childu5	.1230955	.0211052	5.83	0.000	.08173 .164461
expr	.0112089	.0028784	3.89	0.000	.0055674 .0168504
exprsq	-.0001405	.000055	-2.55	0.011	-.0002483 -.0000327
pgradoth	-.0008038	.0125525	-0.06	0.949	-.0254062 .0237986
pracsize	-.0084466	.0122798	-0.69	0.492	-.0325145 .0156214
_cons	4.273103	.4533413	9.43	0.000	3.38457 5.161636
var(e.logyearn)	.1555543	.0129337			.1321625 .1830862

```
. estimates store fmm2
```

The iteration log, suppressed for brevity, shows that optimization uses the expectation-maximization method for the first 20 iterations before switching to a gradient-based algorithm for 4 iterations until convergence is achieved.

The first set of computer output gives estimates of the logit model used to estimate the class probabilities. With just two components, the model is a binary logit model, one that is an intercept-only model because the model does not include individual characteristics. The logit coefficient of 1.1458 implies that the probability of being in the second latent class is  $e^{1.1458}/(1 + e^{1.1458}) = 0.759$ .

The second and third sets of output give the log-linear model estimates for the two latent classes. The coefficients can be interpreted as semielasticities or, for variables that enter in logs, as elasticities. For example, the elasticity with respect to hours worked is much greater in the second class (1.02) than in the first class (0.37). A formal significance test of equality of class-specific coefficients is given later in this section.

More detailed interpretation of results follows.

### 14.3.3 Predicted conditional means in each component

The `fmm` postestimation `predict` command provides predictions for each observation of several quantities. We focus on prediction of the conditional mean (option `mu`), the latent class probabilities (option `classpr`), and the posterior latent probabilities (option `classposteriorpr`). Other options yield predictions of the linear predictor ( $\mathbf{x}'\boldsymbol{\beta}$ ), density, distribution, survival function, and score.

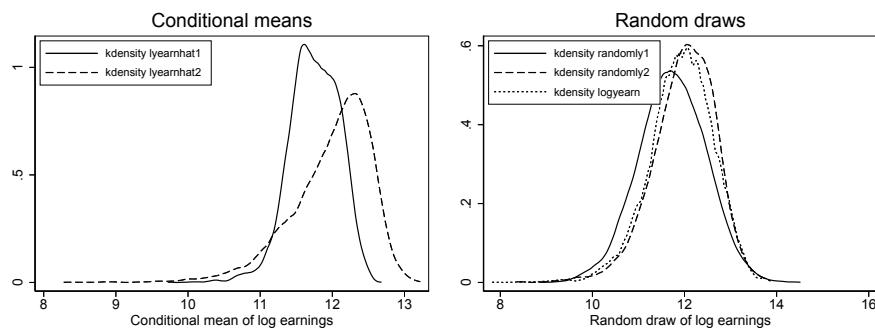
```
. * Two-component mixture: Predicted conditional means for each observation
. predict lyearnhat*, mu
. summarize lyearnhat*
```

Variable	Obs	Mean	Std. dev.	Min	Max
lyearnhat1	5,384	11.74914	.3429049	9.718239	12.68126
lyearnhat2	5,384	11.9832	.5590969	8.277566	13.21821

The suffix `*` leads to prediction for each component. The command `predict lyhat1, mu class(1)`, for example, predicts for just the first component.

The average predicted means for the two components are 11.75 and 11.98. So the second component has a mean that is 0.23 higher on a log scale. The second component also has considerably more variability, with standard deviation 0.56 versus 0.34.

The first panel of figure 14.3 provides a plot of the distribution of the conditional means in the two components. As already stated, the second component has a higher mean and variance than the first component.



**Figure 14.3.** Component-wise density of fitted values

```
. * Distribution of conditional means for the two components
. twoway (kdensity lyearnhat1, lwidth(medthick) clstyle(p1) note(" ")),
>      (kdensity lyearnhat2, lwidth(medthick) clstyle(p2) note(" ")),
>      legend(pos(11) ring(0) col(1)) legend(size(small))
>      xtitle("Conditional mean of log earnings")
>      title("Conditional means")
```

The postestimation command `estat lcmean` reports for each component the sample average predicted conditional mean, along with a standard error and associated 95% confidence interval that controls for estimation error. We have

Latent class marginal means						Number of obs = 5,384
	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1	logyearn	11.74914	.0347218	338.38	0.000	11.68108 11.81719
2	logyearn	11.9832	.0192156	623.62	0.000	11.94553 12.02086

The results 11.75 and 11.98 equal the averages of the predicted conditional means obtained earlier.

#### 14.3.4 Random draws from each fitted component

The individual observations on log-earnings are normally distributed in each component with means given by the preceding `predict, mu` command and variances from the `fmm` prefix output of, respectively, 0.4314 and 0.1555.

To plot the predicted distributions for each component, we make a random draw from the normal distribution with the predicted mean (`lyearnhat1` or `lyearnhat2`) and variance (0.431 or 0.156) that were estimated by the `fmm` prefix. We have

Variable	Obs	Mean	Std. dev.	Min	Max
randomly1	5,384	11.73362	.7318191	8.835665	14.51798
randomly2	5,384	11.9876	.6800995	8.361057	13.86695
<pre>. * Random draws from the two components . generate randomly1 = lyearnhat1 + rnormal(0,sqrt(.4314492)) . generate randomly2 = lyearnhat2 + rnormal(0,sqrt(.1555543)) . summarize randomly1 randomly2</pre>					
<pre>Variable      Obs        Mean       Std. dev.      Min       Max randomly1    5,384    11.73362     .7318191    8.835665   14.51798 randomly2    5,384    11.9876      .6800995    8.361057   13.86695</pre>					
<pre>. twoway (kdensity randomly1, lwidth(medthick) clstyle(p1) note(" ")) &gt;      (kdensity randomly2, lwidth(medthick) clstyle(p2) note(" ")) &gt;      (kdensity logyearn, lwidth(medthick) clstyle(p3) note(" ")), &gt;      legend(pos(11) ring(0) col(1)) legend(size(small)) &gt;      xtitle("Random draw of log earnings") &gt;      title("Random draws")</pre>					

The second panel of figure 14.3 provides a plot of the generated densities of the conditional means. The actual data have distribution that lies between the two components. This example is a case of a mixture that is not well separated and hence does not deviate from unimodality in distribution.

### 14.3.5 Marginal effects

The marginal effects of changes in the regressors on log earnings can be obtained using the `margins` command. The following gives the average marginal effect.

```
. * Two-component mixture: Average marginal effects
. margins, dydx(*)

Average marginal effects                                         Number of obs = 5,384
Model VCE: Robust

Expression: Predicted mean (log of annual earnings), using class probabilities,
            predict(mu outcome(logyearn))
dy/dx wrt:  logyhrs female childu5 expr exprsq pgradoth pracsize
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
logyhrs	.8608154	.0197326	43.62	0.000	.8221402	.8994905
female	-.244768	.0170957	-14.32	0.000	-.278275	-.2112609
childu5	.1149182	.0199822	5.75	0.000	.0757539	.1540826
expr	.0137865	.0025413	5.42	0.000	.0088056	.0187674
exprsq	-.0002676	.0000525	-5.10	0.000	-.0003705	-.0001647
pgradoth	.0245659	.0084442	2.91	0.004	.0080155	.0411162
pracsize	.0211921	.0063143	3.36	0.001	.0088163	.0335679

The average of individual marginal effects for log earnings is similar to the OLS coefficients of the log-linear model given earlier. The average marginal effect of a change in `logyhrs`, the most statistically significant regressor, has increased from 0.804 to 0.861.

Average marginal effects for the first component can be obtained using the command `margins, dydx(*) expression(predict(mu class(1)))`. For the second component, replace `class(1)` with `class(2)`. From output not included, the average marginal-effects vary considerably across the two categories.

Marginal effects generated from a finite mixture of normals will be a weighted average of the component-specific marginal effects. In a linear regression model, this weighted average would be exactly the same as that generated by the standard one-component model. Hence, if the parameter of interest is the aggregate marginal effect, there is no advantage in the mixture model. But if the interest is in group-level heterogeneity of marginal effects, the `fmm` prefix provides a useful framework. This issue is covered in greater detail in sections 23.2 and 23.3.

### 14.3.6 Predicted class probabilities

The `classpr` option of the `predict` command gives the class probabilities  $\pi_{ji}$  for each individual. We obtain

```
. * Two-component mixture: Predicted class probabilities for each observation
. predict classprob*, classpr
. summarize classprob*
```

Variable	Obs	Mean	Std. dev.	Min	Max
classprob1	5,384	.2412507	0	.2412507	.2412507
classprob2	5,384	.7587493	0	.7587493	.7587493

In this example, the class probabilities were specified to not vary with regressors, so they are identical across individuals. They equal 0.241 ( $\pi$ ) and 0.759 ( $1 - \pi$ ), as was manually computed earlier from the `fmm` prefix output. The class sizes are approximately 24% in class 1 and 76% in class 2.

The `estat lcprob` postestimation command reports the sample average predicted class probabilities, along with a standard error and associated 95% confidence interval that controls for estimation error. We have

Latent class marginal probabilities				Number of obs = 5,384
	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.2412507	.0783494	.1208137	.4238662
2	.7587493	.0783494	.5761338	.8791863

The class probabilities are statistically different from 0 and 1 at the 0.05 level, though they are not as precisely estimated as the regression coefficients in each component.

### 14.3.7 Predicted class posterior probabilities

Class contrasts can be carried out also using estimates of the conditional probability of assigning an observation to a particular class. This can be done after estimating for each observation the probability of belonging to each latent class and then finally assigning it to the highest probability class.

Let  $d_{ij}$  be an indicator variable that assumes value 1 if observation  $i$  belongs to class  $j$ ,  $j = 1, \dots, C$ . The probability mass function  $f(d_{ij}|y_i)$  of  $d_{ij}$  conditional on  $y_i$  gives the so-called *posterior probability* that the  $i$ th observation is in class  $j$ , denoted  $\text{post}_{ij}$ , with

$$\begin{aligned}\text{post}_{ij} &= \Pr(\text{observation } i \text{ belongs to class } j | y_i, \mathbf{x}_i) \\ &= \Pr(d_{ij} = 1 | y_i, \mathbf{x}_i) \\ &= \frac{\pi_{ji} f_j(y_i | \mathbf{x}_i, \boldsymbol{\theta}_j)}{\sum_{k=1}^C \pi_{ki} f_k(y_i | \mathbf{x}_i, \boldsymbol{\theta}_k)}\end{aligned}$$

The posterior probabilities are computed using the `predict` command with option `classposteriorpr`, as follows:

```
. * Two-component mixture: Predicted posterior class probabilities for each obs
. predict postprob*, classposteriorpr
. summarize postprob*
```

Variable	Obs	Mean	Std. dev.	Min	Max
postprob1	5,384	.2412507	.1979325	.0046337	1
postprob2	5,384	.7587493	.1979325	3.80e-19	.9953663

Unlike the class probabilities, the posterior probabilities vary considerably across individuals. At the extremes, one individual is viewed as being in the first latent class with probability 1.000, while another is viewed as being in the second latent class component with probability 0.995.

In this example, the posterior probabilities on average equal the class probabilities. This is because the class probabilities do not vary with regressors, so  $\pi_{ji} = \pi_j$ . Some algebra shows that then

$$(1/N) \sum_{i=1}^N \text{post}_{ij} = \pi_j / \sum_{k=1}^C \pi_k = \pi_j.$$

### 14.3.8 Model comparison and selection

To choose between alternative models, one can use the standard approach of choosing the model with the best fit to the data as measured by penalized log likelihood. Akaike's information criterion (AIC) and the Bayesian information criterion (BIC) are commonly used information criteria; see section [13.8.2](#). In Stata, these criteria are defined as follows:

$$\begin{aligned} \text{AIC} &= -2 \ln L + 2K \\ \text{BIC} &= -2 \ln L + K \ln N \end{aligned}$$

Smaller values are preferred. The relevant statistics are obtained using the `estimates stats` postestimation command. While each penalizes a model with more parameters ( $K$ ), the BIC applies a stricter penalty.

To illustrate model comparison and model selection, we fit models with one to four components. Adding an additional component, especially if it is small, may slow down convergence. A common occurrence in such a case is the message “not concave” in the iteration log; this indicates that the algorithm has encountered a segment of log likelihood where concavity fails. This can be a temporary feature in intermediate iterations that disappears; see section 16.3. However, in a highly parameterized model with too many components, it can be a persistent feature that indicates that the model is not identified; see section 16.3.5.

```

. * Estimate one- two- three- and four-component mixture of normals
. qui fmm 1, vce(robust): regress logyearn $xlist
. estimates store fmm1
. qui fmm 2, vce(robust): regress logyearn $xlist
. estimates store fmm2
. qui fmm 3, vce(robust): regress logyearn $xlist
. estimates store fmm3
. estat lcprob

```

Latent class marginal probabilities Number of obs = 5,384

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.3338411	.0333557	.2719343	.4020586
2	.0493479	.015153	.0268224	.0890591
3	.6168109	.0321315	.5522054	.6775385

```

. qui fmm 4, vce(robust): regress logyearn $xlist
. estimates store fmm4
. estat lcprob

```

Latent class marginal probabilities Number of obs = 5,384

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.0194533	.0236252	.0017479	.1835359
2	.3534858	.047536	.2666995	.4511388
3	.0668228	.0317563	.0257136	.1626805
4	.5602382	.0402829	.4804263	.6370522

In the `fmm3` case, observe that the second component is the smallest; furthermore, the output is no longer ordered in terms of the size ranking of the component. That is, the labeling of the components is not the same as in the case of the `fmm2` model. The `fmm4` model has a component with probability of only 0.019. One might be concerned about overfitting, given the standard error of 0.024.

We next compare the models on the basis of information criteria.

```
. * Model comparison: one- two- three- and four-component mixture of normals
. estimates stats fmm1 fmm2 fmm3 fmm4
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
fmm1	5,384	.	-4077.703	9	8173.406	8232.727
fmm2	5,384	.	-3820.455	19	7678.909	7804.142
fmm3	5,384	.	-3708.813	29	7475.626	7666.77
fmm4	5,384	.	-3684.204	39	7446.408	7703.464

Note: BIC uses N = number of observations. See [R] BIC note.

According to the AIC and BIC criteria, fmm3 fits the data better than fmm2. fmm4 does even better on grounds of AIC though not BIC. It is common to emphasize parsimony and use BIC, in which case the fmm3 model is preferred.

### 14.3.9 Tests of equal coefficients

To test whether individual coefficients or subsets of coefficients in different components are significantly different, we can apply a formal test. We illustrate the method by using the fmm2 model to test the equality of a single coefficient and then the joint equality of two coefficients. The first step is to run the model, and then the command fmm, coeflegend is used to collect variable labels.

```
. * Coeflegend gives full names of regression coefficients
. qui fmm 2, vce(robust): regress logyearn logyhrs female childu5
>     expr exprsq pgradoth pracsize
. fmm, coeflegend
Finite mixture model                                         Number of obs = 5,384
Log pseudolikelihood = -3820.4547
```

	Coefficient Legend
1.Class	(base outcome)
2.Class _cons	1.145835 _b[2.Class:_cons]

Class: 1  
 Response: logyearn  
 Model: regress

	Coefficient	Legend
logyearn		
logyhrs	.3654256	_b[logyearn:1.Class#c.logyhrs]
female	-.3316877	_b[logyearn:1.Class#c.female]
childu5	.0892002	_b[logyearn:1.Class#c.childu5]
expr	.0218933	_b[logyearn:1.Class#c.expr]
exprsq	-.0006673	_b[logyearn:1.Class#c.exprsq]
pgradoth	.104355	_b[logyearn:1.Class#c.pgradoth]
pracsize	.1144076	_b[logyearn:1.Class#c.pracsize]
_cons	8.630774	_b[logyearn:1.Class]
var(e.logyearn)	.4314492	_b[/var(e.logyearn)#1.Class]

Class: 2  
 Response: logyearn  
 Model: regress

	Coefficient	Legend
logyearn		
logyhrs	1.018329	_b[logyearn:2.Class#c.logyhrs]
female	-.2171311	_b[logyearn:2.Class#c.female]
childu5	.1230955	_b[logyearn:2.Class#c.childu5]
expr	.0112089	_b[logyearn:2.Class#c.expr]
exprsq	-.0001405	_b[logyearn:2.Class#c.exprsq]
pgradoth	-.0008038	_b[logyearn:2.Class#c.pgradoth]
pracsize	-.0084466	_b[logyearn:2.Class#c.pracsize]
_cons	4.273103	_b[logyearn:2.Class]
var(e.logyearn)	.1555543	_b[/var(e.logyearn)#2.Class]

The next step is to implement the test, here on logyhrs.

```
. * FMM2: Test of a single restriction across latent classes
. test (_b[logyearn:1.Class#c.logyhrs] = _b[logyearn:2.Class#c.logyhrs])
( 1) [logyearn]1bn.Class#c.logyhrs - [logyearn]2.Class#c.logyhrs = 0
      chi2( 1) =    36.32
      Prob > chi2 =    0.0000
```

The single restriction that logyhrs has equal coefficients across the two components is easily rejected.

The identical result is obtained using the contrast command.

```
. * FMM2: Same test using the contrast command
. contrast c.logyhrs#a.Class, equation(logyearn)
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
logyearn Class#c.logyhrs	1	36.32	0.0000

	Contrast	Std. err.	[95% conf. interval]
logyearn Class#c.logyhrs (1 vs 2)	-.652903	.1083366	-.8652388 -.4405673

We next implement a joint test of whether the two variables `female` and `childu5` have equal coefficients across the two classes.

```
. * FMM2: Test of a joint restriction across latent classes
. test (_b[logyearn:1.Class#c.female] = _b[logyearn:2.Class#c.female])
>      (_b[logyearn:1.Class#c.childu5]= _b[logyearn:2.Class#c.childu5])
( 1) [logyearn]1bn.Class#c.female - [logyearn]2.Class#c.female = 0
( 2) [logyearn]1bn.Class#c.childu5 - [logyearn]2.Class#c.childu5 = 0
      chi2( 2) =     1.66
      Prob > chi2 =    0.4365
```

This joint restriction has much higher *p*-value (0.437) and is not rejected at, for example, the 0.05 level.

Overall significant differences across classes can arise from a small number of key differences. Section 23.3.2 shows how even greater flexibility in specifying mixture components can be attained by placing exclusion restrictions on them.

### 14.3.10 More flexible FMM models

In some cases, it may be desirable to relax the latent class restriction that the component weights are constant. The additional flexibility comes at a cost. The modeler will need to offer reasons for including a regressor in the

probability function rather than the mean function. Sometimes, the same variable may appear in both functions. To illustrate the dilemma, we estimate a simplified variant of the  $\text{fmm2}$  model considered earlier.

```

. * FMM2 model with varying mixture probabilities
. fmm 2, lcprob(female) lcbase(2) nolog vce(robust):
>      regress logyearn logyhrs expr exprsq pgradoth pracsize
Finite mixture model                                         Number of obs = 5,384
Log pseudolikelihood = -4014.1217

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1. Class						
female	3.229546	.3096566	10.43	0.000	2.62263	3.836461
_cons	-.1219929	.237477	-0.51	0.607	-.5874393	.3434535
2. Class	(base outcome)					

Class: 1  
 Response: logyearn  
 Model: regress

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
logyearn						
logyhrs	.8901694	.0277857	32.04	0.000	.8357105	.9446283
expr	.0001189	.0036869	0.03	0.974	-.0071074	.0073451
exprsq	-.000094	.0000867	-1.08	0.278	-.0002639	.0000759
pgradoth	.0346003	.0124546	2.78	0.005	.0101898	.0590108
pracsize	.0492521	.0089879	5.48	0.000	.0316361	.0668681
_cons	4.955431	.2040458	24.29	0.000	4.555509	5.355354
var(e.logyearn)	.2370821	.0124561			.2138834	.2627969

Class: 2  
 Response: logyearn  
 Model: regress

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
logyearn						
logyhrs	.5326659	.0889074	5.99	0.000	.3584105	.7069212
expr	.03665	.0064563	5.68	0.000	.0239959	.0493041
exprsq	-.0006263	.0001183	-5.29	0.000	-.0008581	-.0003944
pgradoth	.0146541	.0183619	0.80	0.425	-.0213345	.0506427
pracsize	-.042831	.0143324	-2.99	0.003	-.0709221	-.01474
_cons	8.004132	.6879643	11.63	0.000	6.655747	9.352517
var(e.logyearn)	.1710046	.0107656			.1511543	.1934618

In this model specification, the mixing weight  $\pi$  is a function of a binary indicator of gender (`female`). The heuristic notion is that being female may lower the probability of being in a higher average income class perhaps because of gender-related wage discrimination. In this example, we treat the higher income group, class 2, as the base group.

The estimates show that being female raises the probability of being in class 1. However, note that the interpretation of results will be more complex if our specification also uses `female` as a regressor in the mean function and, further, adds other variables such as `childus5`. Unfortunately, it may not be easy to get clear guidance on how different factors affect the outcome, whether through mixing proportions, or through the conditional mean, or both. Hence, an improvement in the fit of the model resulting from parameterizing  $\pi$  can be interpreted as resulting from the use of a more flexible functional form. For further discussion of related issues, see section 23.3.

## 14.4 Global polynomials

Most regression models, either models of the conditional mean or fully parametric models, enter regressors as a linear combination of the form  $\mathbf{x}'\beta$ .

A linear model may appear restrictive. It is linear in the parameters  $\beta$ . Nonlinearity can be introduced, however, by setting the regressors  $\mathbf{x}$  to be nonlinear transformations of the underlying variables of interest.

In this section, we use global polynomials that fit a single polynomial model throughout the range of the regressors. In the subsequent section, we use regression splines that fit separate polynomials over subranges of  $x$  in such a way that these separate polynomials are connected (or splined) and lead to a continuous function for  $E(y|x)$ .

Polynomials and splines can be included as regressors in any type of regression, not just least-squares regression. In the special case of least-squares regression, the `npregress series` command can be used to fit polynomial and spline models. This command is deferred to section [14.6.6](#) and presented in further detail in chapter 27.

### 14.4.1 Global polynomials

The following dataset generates dependent variable  $y$  as a function of the correlated variables  $x_1$ ,  $x_2$ ,  $z$ , and  $z_{sq}$ .

```
. * Generated data: y = 1 + 1*x1 + 1*x2 + f(z) + u where f(z) = z + z^2
. clear
. set obs 200
Number of observations (_N) was 0, now 200.
. set seed 10101
. generate x1 = rnormal()
. generate x2 = rnormal() + 0.5*x1
. generate z = rnormal() + 0.5*x1
. generate zsq = z^2
. generate y = 1 + x1 + x2 + z + zsq + 2*rnormal()
```

```
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
x1	200	.0301211	1.014172	-3.170636	3.093716
x2	200	.0226274	1.158216	-4.001105	3.049917
z	200	.0664539	1.146429	-3.386704	2.77135
zsq	200	1.312145	1.658477	.0000183	11.46977
y	200	2.164401	3.604061	-5.468721	14.83116

We consider regression of  $y$  on a polynomial function of  $z$ . Because the omitted variables  $x_1$  and  $x_2$  are correlated with  $z$ , the model with  $x_1$  and  $x_2$  omitted is not necessarily quadratic in  $z$ . We fit a quartic model in  $z$ , using factor-variable notation.

```
. * Quartic global polynomial model
. reg y c.z##c.z##c.z##c.z, vce(robust)
```

```
Linear regression
Number of obs      =      200
F(4, 195)          =     96.68
Prob > F          =     0.0000
R-squared           =     0.4889
Root MSE            =     2.6028
```

y	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
z	1.398768	.2765072	5.06	0.000	.8534389	1.944096
c.z#c.z	.8034603	.2273094	3.53	0.001	.3551597	1.251761
c.z#c.z#c.z	.0918065	.0554654	1.66	0.099	-.0175826	.2011957
c.z#c.z#c.z#c.z	.0265145	.0274171	0.97	0.335	-.0275577	.0805866
_cons	.8862917	.2658742	3.33	0.001	.3619334	1.41065

The cubic and quartic terms turn out to be statistically insignificant at level 0.05.

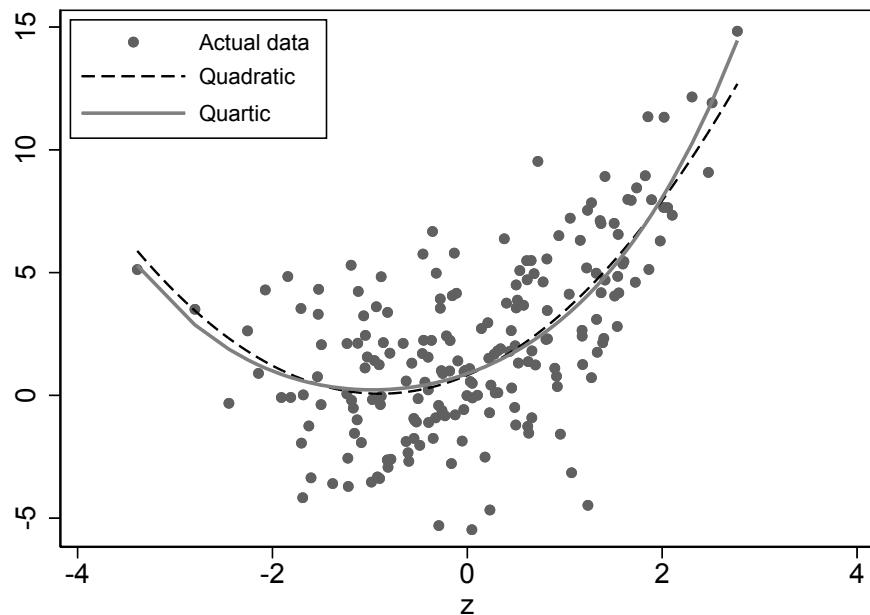
The following code generates a figure that plots predictions from this quartic model and compares these to a plot of predictions from a quadratic model obtained directly using the `qfit` graphics command.

```

. * Graph comparing quartic model predictions to quadratic model predictions
. predict yquartic, xb
. sort z
. twoway (scatter y z, msize(small)) (qfit y z, lwidth(medthick)clstyle(p2))
>     (line yquartic z, lwidth(medthick)), scale(1.2)
>     legend(pos(11) ring(0) col(1)) legend(size(small))
>     legend(label(1 "Actual data") label(2 "Quadratic") label(3 "Quartic"))

```

Figure 14.4 shows that the quartic overfits by seeking to predict well at the extreme values of  $z$ .



**Figure 14.4.** Plot of fitted quadratic and quartic models

#### 14.4.2 Fractional polynomials and orthogonal polynomials

Fractional polynomials allow polynomials to be raised to noninteger powers. A simple example is  $y = \beta_1 + \beta_2 x^{1/2}$ .

The `fp` prefix fits such models. As an example, we refit the quartic model, in which the powers are actually the integer values 1, 2, 3, and 4. The `fp` prefix requires that the regressor be positive, so we use the `scale` option that appropriately transforms the regressor. We have

```
. * Quartic model estimated using fractional polynomial command
. fp <z>, fp(1 2 3 4) scale replace: regress y <z>
-> regress y z_1 z_2 z_3 z_4
```

Source	SS	df	MS	Number of obs	=	200
Model	1263.81882	4	315.954706	F(4, 195)	=	46.64
Residual	1321.04322	195	6.77458062	Prob > F	=	0.0000
Total	2584.86204	199	12.9892565	R-squared	=	0.4889
				Adj R-squared	=	0.4784
				Root MSE	=	2.6028

y	Coefficient	Std. err.	t	P> t	[95% conf. interval]
z_1	-5.004738	3.621983	-1.38	0.169	-12.14803 2.138552
z_2	1.695515	2.055445	0.82	0.410	-2.358242 5.749272
z_3	-.2673974	.469694	-0.57	0.570	-1.19373 .658935
z_4	.0265145	.0369361	0.72	0.474	-.0463311 .09936
_cons	5.287323	2.307045	2.29	0.023	.7373604 9.837286

```
. predict yfpquartic
(option xb assumed; fitted values)
. correlate yquartic yfpquartic
(obs=200)
```

	yquartic yfpqua^c	
yquartic	1.0000	
yfpquartic	1.0000	1.0000

Because of the rescaling of  $z$ , the coefficient estimates differ from those given earlier for the quartic model, aside from that for the fourth-order term. But the model fit is the same and yields the same predicted values. The `fp` prefix can also include  $\ln x$  and interactions with  $\ln x$  as regressors. And the `mfp` prefix extends to polynomials in more than one variable.

The terms in a polynomial can be highly collinear, leading in some cases to numerical problems in computing an estimator. The `orthpoly` command creates, for integer-powered polynomials, a linear transformation that creates a set of polynomials that are uncorrelated with each other.

For example, to transform the quartic polynomials for  $z$  to uncorrelated polynomials, we give the command

```

. * Orthogonalize the quartic polynomials
. orthpoly z, generate(pz*) deg(4)
. correlate pz*
(obs=200)

```

	pz1	pz2	pz3	pz4
pz1	1.0000			
pz2	-0.0000	1.0000		
pz3	0.0000	0.0000	1.0000	
pz4	0.0000	0.0000	-0.0000	1.0000

The command `regress y pz*` will then lead to the same fitted values of  $y$  as the initial quartic regression of  $y$  on the first four powers of  $z$ .

## 14.5 Regression splines

Spline regression, or piecewise regression, breaks the range of  $x$  into a few segments, for example, five segments, and fits a separate polynomial regression within each segment. This is done in a way that ensures that the predictions coincide at the segment boundaries, so that the predictions are continuous in the regressor. Leading examples are piecewise linear regression and cubic splines. While the discussion here considers only OLS regression, the splines can be used in more general models such as in logistic regression.

Nonparametric estimation of these models using the `npregress series` command is deferred to section [14.6.6](#).

### 14.5.1 Piecewise linear regression

We begin with piecewise linear regression with the range of  $x$  broken into three segments:  $(-\infty, c)$ ,  $[c, d)$ , and  $[d, \infty)$ . The segment boundaries  $c$  and  $d$  are called **knots** and are specified by the researcher rather than parameters to be estimated.

Then  $y = f(x) + u$ , where

$$f(x) = \mathbf{1}(x < c)(\alpha_1 + \alpha_2x) + \mathbf{1}(c \leq x < d)(\alpha_3 + \alpha_4x) + \mathbf{1}(x \geq d)(\alpha_5 + \alpha_6x)$$

where  $\mathbf{1}(A) = 1$  if event  $A$  occurs and  $\mathbf{1}(A) = 0$  otherwise.

If this regression is run, then the three separate lines will not connect at the knots  $c$  and  $d$ . To ensure continuity at  $c$ , so  $f(c-) = f(c)$ , we need  $\alpha_1 + \alpha_2c = \alpha_3 + \alpha_4c$ . And to ensure continuity at  $d$ , so  $f(d-) = f(d)$ , we need  $\alpha_3 + \alpha_4d = \alpha_5 + \alpha_6d$ .

With some algebra, it can be shown that these two constraints are met if we define

$$f(x) = \beta_0 + \beta_1x + \beta_2(x - c)_+ + \beta_3(x - d)_+ \tag{14.1}$$

where  $(\cdot)_+$  denotes the Heaviside function. For example,

$$(x - c)_+ = \mathbf{1}(x - c \geq 0) \times (x - c) = \begin{cases} x - c & \text{if } x > c \\ 0 & \text{otherwise} \end{cases}$$

Four variables (including the intercept) appear in (14.1) because two constraints were imposed on the original unconstrained model for  $f(x)$ . An end-of-chapter exercise demonstrates that (14.1) yields the desired piecewise linear and continuous function in the simpler case of one knot.

We apply this method for regression of  $y$  on  $z$  using the same data as those used in the global polynomials example.

We specify three segments with knots at  $-1$  and  $1$  and create three regressors  $zseg1-zseg3$  corresponding to  $x$ ,  $(x - c)_+$ , and  $(x - d)_+$  in (14.1).

```
. * Create the basis function manually with three segments and knots at -1 and 1
. generate zseg1 = z
. generate zseg2 = 0
. replace zseg2 = z - (-1) if z > -1
(163 real changes made)
. generate zseg3 = 0
. replace zseg3 = z - 1 if z > 1
(47 real changes made)
```

Regression of  $y$  on an intercept and  $zseg1-zseg3$ , and subsequent prediction, yields

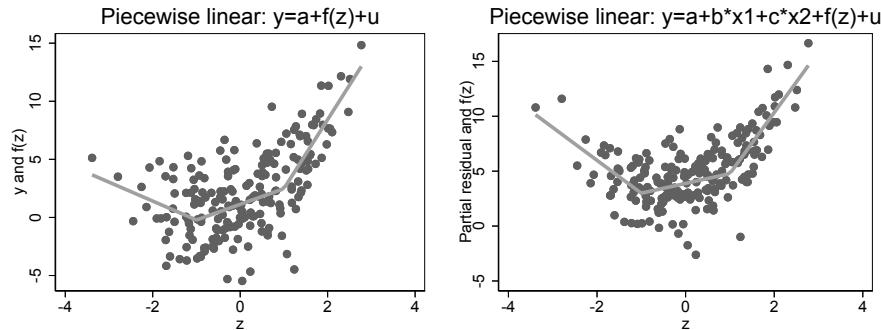
```
. * Piecewise linear regression with three sections
. regress y zseg1 zseg2 zseg3, vce(robust)

Linear regression
Number of obs      =      200
F(3, 196)          =     97.11
Prob > F          =     0.0000
R-squared           =     0.4849
Root MSE            =     2.6064
```

y	Robust					[95% conf. interval]
	Coefficient	std. err.	t	P> t		
zseg1	-1.629491	.5128884	-3.18	0.002	-2.64098	-.618003
zseg2	2.977586	.7302596	4.08	0.000	1.537411	4.417761
zseg3	4.594974	.855389	5.37	0.000	2.908026	6.281922
_cons	-1.850531	.7809994	-2.37	0.019	-3.390772	-.3102895

```
. predict yhat
(option xb assumed; fitted values)
. twoway (scatter y z) (line yhat z, sort lwidth(thick)),
>       title("Piecewise linear: y=a+f(z)+u") ytitle("y and f(z)") xtitle("z")
>       legend(off)
```

The first panel of figure 14.5 plots  $\hat{f}(z)$ , the predicted value of  $y$ , against  $z$ . As expected, there are separate lines in the three segments, and the lines are connected.



**Figure 14.5.** Piecewise linear: Single regressor  $z$  without and with additional regressors

The `mkspline` command, with the `marginal` option, creates the same linear spline variables. We have

```

. * Repeat piecewise linear using command mkspline to create the basis functions
. mkspline zmk1 -1 zmk2 1 zmk3 = z, marginal
. summarize zseg1 zmk1 zseg2 zmk2 zseg3 zmk3, sep(8)

```

Variable	Obs	Mean	Std. dev.	Min	Max
zseg1	200	.0664539	1.146429	-3.386704	2.77135
zmk1	200	.0664539	1.146429	-3.386704	2.77135
zseg2	200	1.171111	.984493	0	3.77135
zmk2	200	1.171111	.984493	0	3.77135
zseg3	200	.138441	.3169973	0	1.77135
zmk3	200	.138441	.3169973	0	1.77135

```

. regress y zmk1 zmk2 zmk3, vce(robust) noheader

```

y	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
zmk1	-1.629491	.5128884	-3.18	0.002	-2.64098	-.618003
zmk2	2.977586	.7302596	4.08	0.000	1.537411	4.417761
zmk3	4.594974	.855389	5.37	0.000	2.908026	6.281922
_cons	-1.850531	.7809994	-2.37	0.019	-3.390772	-.3102895

The output confirms that the `mkspline` command creates the same variables and leads to the same regression results.

We now include additional regressors  $x_1$  and  $x_2$  and verify that the resulting prediction  $\hat{f}(z)$  remains a piecewise linear and continuous function of  $z$ .

```
. * Piecewise regression with additional regressors x1 and x2
. regress y x1 x2 zmk1 zmk2 zmk3, vce(robust)

Linear regression
Number of obs = 200
F(5, 194) = 110.06
Prob > F = 0.0000
R-squared = 0.6830
Root MSE = 2.0551
```

y	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
x1	.8776081	.1582844	5.54	0.000	.5654288	1.189787
x2	.987047	.1327458	7.44	0.000	.7252369	1.248857
zmk1	-2.993494	.4841391	-6.18	0.000	-3.948346	-2.038642
zmk2	3.886034	.6262328	6.21	0.000	2.650936	5.121133
zmk3	4.665117	.6898819	6.76	0.000	3.304485	6.025748
_cons	-2.882261	.6715345	-4.29	0.000	-4.206706	-1.557815

```
. generate partresid = y - _b[_cons] - _b[x1]*x1 - _b[x2]*x2
. generate fz = _b[zmk1]*zmk1 + _b[zmk2]*zmk2 + _b[zmk3]*zmk3

. twoway (scatter partresid z) (line fz z, sort lwidth(thick)), xtitle("z")
> title("Piecewise linear: y=a+b*x1+c*x2+f(z)+u") legend(off)
> ytitle("Partial residual and f(z)")
```

The second panel of figure 14.5 confirms that  $\hat{f}(z)$  is a piecewise linear and continuous function of  $z$ .

In general, if a piecewise linear and continuous function  $f(x)$  is desired for a regressor  $x$  and there are  $K$  knots at  $c_1, \dots, c_K$ , then

$$f(x) = f(x) = \beta_0 + \beta_1 x + \beta_2(x - c_1)_+ + \dots + \beta_{K+1}(x - c_K)_+$$

A common procedure is to use equal numbers of observations in each segment. For example, with three knots, set the knots at the 25th, 50th, and 75th percentiles of  $x$ .

### 14.5.2 Natural or restricted cubic splines

The piecewise linear example can be extended to higher-order polynomials. A common choice is a cubic spline because it is the lowest-degree regression

spline where the graph of  $\hat{f}(x)$  on  $x$  appears to be smooth and continuous to the naked eye.

For a cubic spline with  $K$  knots, we require  $f(x)$ ,  $f'(x)$ , and  $f''(x)$  to be continuous at the  $K$  knots. This imposes additional constraints to those in the linear case. It can be shown that we can do OLS with

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - c_1)_+^3 + \cdots + \beta_{(3+K)} (x - c_K)_+^3$$

An end-of-chapter exercise verifies a similar result in the simpler case of a quadratic spline with one knot. The variables in the preceding expression are called basis functions.

In the cubic case with  $K$  knots, there are  $(K + 4)$  basis functions, including the intercept. A limitation, however, is that the cubic spline overfits at the boundaries of the data.

A natural spline or restricted spline is an adaptation that restricts the relationship to be linear past the lower and upper knots. This reduction from cubic to linear imposes four restrictions—two at each endpoint—so in the cubic case with  $K$  knots, there are  $K$  basis functions, including the intercept.

Natural or restricted cubic splines can be created using the `cubic` option of the `mkspline` command. We consider an example with five knots. Using the `knots(5)` option, we see the default to the five knots at the 5, 27.5, 50, 72.5, and 95 percentiles of the regressor. The smallest and largest knots are specified to be relatively close to the boundaries of the data, so that the linear restriction applies to a relatively small amount of the data.

With five knots, we obtain

```

. * Natural or restricted cubic spline regression of y on z
. mkspline zspline = z, cubic nknots(5) displayknots

```

	knot1	knot2	knot3	knot4	knot5
z	-1.707005	-.6282565	.0096633	.8139451	1.889486

```

. regress y zspline*, vce(robust)

Linear regression
Number of obs      =       200
F(4, 195)          =      72.39
Prob > F           =     0.0000
R-squared           =     0.4846
Root MSE            =     2.6138

```

y	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
zspline1	-1.577777	.5393653	-2.93	0.004	-2.641515	-.5140385
zspline2	8.974681	3.677336	2.44	0.016	1.722224	16.22714
zspline3	-32.6592	20.52346	-1.59	0.113	-73.13566	7.817256
zspline4	46.24063	31.10605	1.49	0.139	-15.10685	107.5881
_cons	-1.745086	.8845348	-1.97	0.050	-3.489569	-.0006031

The knots correspond to  $z$  values of  $-1.707, \dots, 1.889$ .

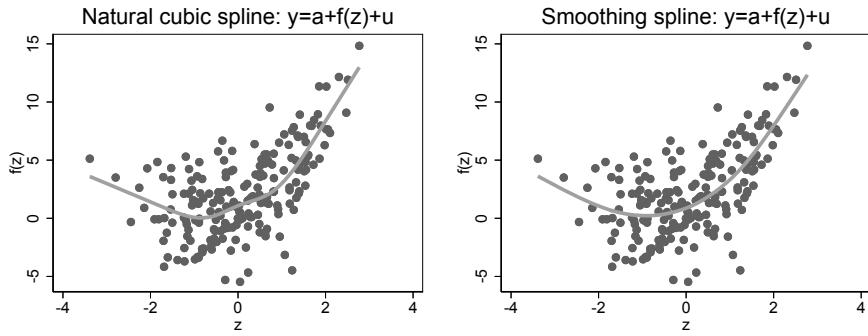
We then create a plot of the fitted values against  $z$ .

```

. * Plot the predicted values from natural cubic spline regression
. predict yhatnatural
(option xb assumed; fitted values)
. twoway (scatter y z) (line yhatnatural z, sort lwidth(thick)),
>      title("Natural cubic spline:  $y=a+f(z)+u$ ") xtitle("z")
>      ytitle("f(z)") legend(off)

```

The first panel in figure 14.6 displays a linear relationship for values of  $z$  less than the smallest knot ( $-1.71$ ) and greater than the largest knot ( $1.89$ ) and a nonlinear relationship between these two values.



**Figure 14.6.** Natural cubic spline and smoothing spline

### 14.5.3 Smoothing splines

The preceding splines are special cases of linear basis expansions in the regressor, with

$$f(x) = \sum_{m=1}^M \beta_m h_m(x)$$

where  $h_m(x)$ ,  $m = 1, \dots, M$  are  $M$  functions of  $x$ .

Regression splines and natural splines require specifying the knots. Smoothing splines use all distinct values of  $x$  as knots but then add a smoothness penalty that penalizes curvature. The function  $f(\cdot)$  minimizes

$$\sum_{i=1}^n \{y_i - f(x_i)\}^2 + \lambda \int f''(t) dt$$

The parameter  $\lambda$  determines the amount of smoothing. When  $\lambda = 0$ , the fitted  $f(x_i)$  connects the data points, while  $\lambda = \infty$  yields OLS.

The `gam` command ([Royston and Ambler 1998](#)), presented in section 27.8, uses smoothing splines. We implement this for regression of  $y$  on  $z$ . The

degree of smoothing is determined by the effective degrees of freedom, which are inversely related with  $\lambda$ . We set the effective degrees of freedom to 4 to impose similar flexibility to the preceding natural cubic spline regression that included 4 regressors aside from the intercept. We have

```
. * Smoothing spline regression of y on z - requires Windows version of Stata
. gam y z, df(4)
200 records merged.

Generalized Additive Model with family gauss, link ident.

Model df      =      5.006
Deviance     =    1329.55
                                         No. of obs =      200
                                         Dispersion =  6.81841



| y     | df    | Lin. Coef. | Std. Err. | z      | Gain   | P>Gain |
|-------|-------|------------|-----------|--------|--------|--------|
| z     | 4.006 | 1.718039   | .1614613  | 10.641 | 70.867 | 0.0000 |
| _cons | 1     | 2.1644     | .18464    | 11.722 | .      | .      |


Total gain (nonlinearity chisquare) = 70.867 (3.006 df), P = 0.0000
. display "R2 = " 1 - e(disp)*e(tdf) / 2584.6
R2 = .48558808
. twoway (scatter y z) (line GAM_mu z, sort lwidth(thick)),
>      title("Smoothing spline: y=a+f(z)+u") xtitle("z")
>      ytitle("f(z)") legend(off)
```

The  $R^2$  is essentially the same as that for the natural cubic spline defined in section [14.5.2](#).

The second panel of figure [14.6](#) plots the fitted values from the smoothing spline regression against  $z$ . The fitted curve is very close to that from natural cubic spline regression.

Other basis functions have been proposed, including wavelets and B-splines. The `bspline(#)` option of the `npregress series` command uses B-splines, and the community-contributed `bspline` command ([Newson 2012](#)) enables generation of a range of bases of splines.

## 14.6 Nonparametric regression

Spline regression or piecewise regression breaks the range of  $x$  into a few segments, for example, five segments, and fits a separate polynomial regression within each segment.

Nonparametric regression methods similarly perform a number of separate regressions, but they use many more regressions. Specifically,  $y$  is predicted at many values  $x_0$  of  $x$  using local regression that places greatest weight on data points in the neighborhood of  $x_0$ .

To enable decent prediction when there are few data points, one can base the local regression on a large neighborhood of  $x_0$ , one much wider than the distance between successive evaluation points  $x_0$ . So many overlapping weighted regressions are used.

The methods are called nonparametric because under suitable assumptions, they provide consistent estimates of the conditional mean function  $E(y|x)$  as  $N \rightarrow \infty$ , regardless of the true functional form for  $E(y|x)$ .

Plots based on nonparametric regression were presented in section [2.6.6](#). We repeat some of that material here and present further detail. The focus is on kernel-weighted polynomial regression for a single regressor. Extension to multiple regressors is given in chapter 27.

### 14.6.1 Local regression

Consider the regression model  $y = m(x) + u$ , where  $x$  is a scalar and the conditional mean function  $m(\cdot)$  is not specified. The goal is to estimate  $m(\cdot)$ . A local regression estimate of  $m(x)$  at  $x = x_0$  is a local weighted average of  $y_i$ ,  $i = 1, \dots, N$ , that places great weight on observations for which  $x_i$  is close to  $x_0$  and little or no weight on observations for which  $x_i$  is far from  $x_0$ . Formally,

$$\hat{m}(x_0) = \sum_{i=1}^N w(x_i, x_0, h) y_i$$

where the weights  $w(x_i, x_0, h)$  sum over  $i$  to one and decrease as the distance between  $x_i$  and  $x_0$  increases. More weight is placed on observations for which  $x_i$  is close to  $x_0$  as the parameter  $h$ , called a bandwidth parameter, increases.

The prediction  $\hat{m}(x_0)$  is evaluated at a range of values of  $x_0$ ; an obvious choice is to do so at each distinct value taken by  $x$ . It may seem that there are then too few local data points available to obtain a decent fit. This is avoided by using weights  $w(x_i, x_0, h)$  that average over much more than a small fraction of the data. Essentially, rolling windows are used to evaluate  $\hat{m}(x_0)$  at a range of values of  $x_0$ .

### 14.6.2 Nearest-neighbors regression

The  $k$ -nearest-neighbors estimator uses just the  $k$  observations on  $y_i$  for which  $x_i$  is closest to  $x_0$  and equally weights these  $k$  values of  $y_i$ .

This estimator can be obtained using the community-contributed `knnreg` command ([Salgado-Ugarte, Shimizu, and Taniuchi 1996](#)).

The  $k$ -nearest-neighbors estimator is most often used as a matching method for discriminant analysis and for treatment-effects estimation.

### 14.6.3 Local polynomial regression

Kernel regression at  $x = x_0$  uses the weight

$$w(x_i, x_0, h) = \frac{K\{(x_i - x_0)/h\}}{\sum_{i=1}^N K\{(x_i - x_0)/h\}}$$

where  $K(\cdot)$  is a kernel function defined in section [2.6.4](#) and detailed in section 27.2.3. For example, the `kernel(epan2)` option sets  $K(z) = (3/4)(1 - z^2)$  if  $|z| < 1$  and  $K(z) = 0$  otherwise.

The kernel regression estimate at  $x = x_0$  can equivalently be obtained by minimizing

$$\sum_{i=1}^N w(x_i, x_0, h) \times (y_i - \alpha_0)^2$$

which is weighted regression on a constant where the kernel weights are largest for observations with  $x_i$  close to  $x_0$ . Then  $\hat{m}(x_0) = \hat{\alpha}_0$ . This estimator is also called the (kernel-weighted) local constant estimator.

The (kernel-weighted) local linear estimator of  $m(\cdot)$  additionally includes a slope coefficient and at  $x = x_0$  minimizes

$$\sum_{i=1}^N w(x_i, x_0, h) \times \{y_i - \alpha_0 - \beta_0(x_i - x_0)\}^2 \quad (14.2)$$

Again,  $\hat{m}(x_0) = \hat{\alpha}_0$ . This estimator has the advantage of better estimation of  $m(x_0)$  at values of  $x_0$  near the endpoints of the range of  $x$  because it allows for any trends near the endpoints.

More generally, the local polynomial estimator of degree  $p$  uses a polynomial of degree  $p$  in  $(x_i - x_0)$  in [\(14.2\)](#).

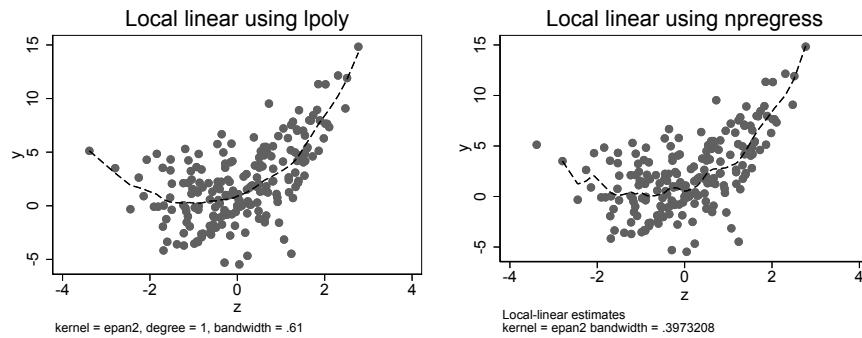
#### 14.6.4 Local linear regression using `lpoly` and `npregress kernel`

The `lpoly` command was introduced in section [2.6.6](#). We use it to perform local linear regression of `y` on `z`. The `at(z)` option is used to obtain  $\hat{m}(z)$  at each sample value of `z`, and the `generate()` option is used to store the

predictions in variable `yhatlpoly`. We use the `epan2` kernel with default bandwidth determined by a plugin formula.

```
. * Local linear using lpoly
. lpoly y z, degree(1) at(z) generate(yhatlpoly) kernel(epan2)
>      title("Local linear using lpoly")
```

The first panel in figure 14.7 presents a plot of the data and the predictions  $\hat{m}(z)$ . The conditional mean function is similar to a quadratic function.



**Figure 14.7.** Local linear regression using `lpoly` and `npregress kernel`

An alternative command is the `npregress kernel` command, which is presented in detail in chapter 27. This command fits local constant and local linear models. It uses cross-validation (see section 27.2.4) to obtain the bandwidth parameter, rather than a plugin formula.

The `npregress kernel` command for local linear regression of `y` on `z` yields the following output:

```
. * Local linear using npregress kernel
. npregress kernel y z, estimator(linear) kernel(epan2) nolog
>     vce(bootstrap, seed(10101) reps(400) nodots)
```

Bandwidth

	Mean	Effect
z	.3973208	.539756

Local-linear regression	Number of obs	=	199
Kernel : epan2	E(Kernel obs)	=	79
Bandwidth: cross-validation	R-squared	=	0.5057

	Observed estimate	Bootstrap std. err.	z	P> z	Percentile [95% conf. interval]	
Mean y	2.165175	.2506887	8.64	0.000	1.671694	2.638386
Effect z	2.009928	.3806551	5.28	0.000	1.286672	2.869827

Note: Effect estimates are averages of derivatives.

```
. npgraph, title("Local linear using npregress")
```

The output reports the sample mean of the predictions  $\hat{m}(z_i)$ . As explained in section 27.2, fitting a local linear model, rather than a local constant model, enables one to also estimate the marginal effects  $\hat{m}'(z_i)$ . The output also reports the sample mean of the marginal effects  $\hat{m}'(z_i)$ . And the output also presents the bandwidths used to estimate  $m(z_i)$  and  $m'(z_i)$ .

The second panel in figure 14.7, obtained using the `npgraph` postestimation command, presents a plot of the data and the predictions  $\hat{m}(z)$ . The fitted curve is more variable because the bandwidth used by the `npregress kernel` command, one determined by cross-validation, is smaller than the bandwidth used by the `lpoly` command, one determined by a plugin formula. The `npregress kernel` command used 199 observations rather than the original 200 observations because local linear methods require at least two observations within the interval defined by the bandwidth, and this was not possible for the smallest value of  $z$ , as can be seen in the second panel of the figure.

## 14.6.5 lowess

The locally weighted scatterplot smoothing estimator (lowess), which we introduced in section [2.6.6](#), is a variation of the local linear estimator. Lowess uses a variable bandwidth, a tricubic kernel, and downweights observations with large residuals (using a method that greatly increases the computational burden).

We obtain predictions from the preceding `npregress` command and from the `lowess` command and compare these with predictions from the `lpoly` command and the actual data on `y`.

```
. * Compare predicted conditional means from lpoly, npregress, and lowess
. predict yhatnp, mean                                // Prediction from npregress
(1 missing value generated)
. lowess y z, generate(yhatlowess)
. sum y yhatnp yhatlpoly yhatlowess
```

Variable	Obs	Mean	Std. dev.	Min	Max
y	200	2.164401	3.604061	-5.468721	14.83116
yhatnp	199	2.165175	2.566053	.0537626	14.90781
yhatlpoly	200	2.230832	2.538389	.2027314	14.27889
yhatlowess	200	2.470852	2.442373	.4760757	13.25927

```
. correlate y yhatnp yhatlpoly yhatlowess
(obs=199)
```

	y	yhatnp	yhatlp^y	yhatlo^s
y	1.0000			
yhatnp	0.7112	1.0000		
yhatlpoly	0.7020	0.9962	1.0000	
yhatlowess	0.6990	0.9920	0.9976	1.0000

The output shows the greater variability in the predictions from `npregress` kernel compared with `lpoly`, due to the smaller bandwidth. All three predictions are highly correlated with each other. The correlation of the `lpoly` predictions with `y` corresponds to  $R^2 = 0.7020^2 = 0.493$ , compared with 0.485 using a natural cubic spline with 5 knots and 0.489 using a quartic global polynomial.

## 14.6.6 Series estimation using `npregress` series

Models with global polynomials of specified degree were introduced in section [14.4](#), and regression splines with specified knots were introduced in section [14.5](#).

These models can be fit using the `npregress series` command, which is presented in detail in section 27.3. The polynomial degree and the number of knots for splines can be specified, or they can be data determined using cross-validation (see section 27.2.4) or information criterion measures.

The following code fits a fourth-degree polynomial model and a third-order regression spline with two knots.

```
. * Examples of the npregress series command with specified degree and knots  
. npregress series y z, polynomial(4)  
    (output omitted)  
. npregress series y z, spline knots(2)  
    (output omitted)
```

More generally, the polynomial degree and the number of knots may be data determined. The default is to determine these using cross-validation.

```
. * Examples of the npregress series command with data-determined degree and knots  
. npregress series y z, polynomial  
    (output omitted)  
. npregress series y z, spline  
    (output omitted)
```

## 14.7 Partially parametric regression

Extending nonparametric regression to a general  $K$ -dimensional regression is difficult because of the “curse of dimensionality”, which essentially implies that in a high-dimensional regression, we can expect to encounter many empty subspaces, regions with no observations, unless the sample size is very large. Then we have an insufficient number of observations to fit a nonparametric regression. The required sample size increases exponentially with the dimension, so nonparametric regression is generally not practicable for relationships with many covariates.

The fully nonparametric approach can be used if one has just one or a few regressors, which is often the situation in which exploratory regressions are used. Chapter 27 presents the `npregress` command in considerable detail when there are multiple regressors.

Alternatively, one might take a semiparametric approach that applies nonparametric methods to part of the model, while other parts of the model are parametric.

Leading examples of semiparametric models are the following. A partially linear model specifies the conditional mean to be the usual linear regression function plus an unspecified nonlinear component, so  $E(y|\mathbf{x}, z) = \mathbf{x}'\boldsymbol{\beta} + g(z)$ . A single-index model specifies the conditional mean to be an unspecified function of a linear combination of the regressors, so  $E(y|\mathbf{x}) = g(\mathbf{x}'\boldsymbol{\beta})$ , where the function  $g(\cdot)$  is unspecified. And a generalized additive model specifies the conditional mean function of a model with  $K$  regressors to be the sum of  $K$  separate unspecified functions for each regressor.

Chapter 27 details these semiparametric models and presents community-contributed programs that fit these models.

## 14.8 Additional resources

The official Stata `fmm` prefix for fitting finite mixture regression models is a successor to the community-contributed `fmm` command by [Deb \(2007\)](#), which was used in the previous edition of this book. The official Stata version fits a more comprehensive suite of models. For example, it can also estimate mixtures of lognormal, Student's  $t$ , gamma, Poisson, and negative binomial regressions.

Regression splines are presented in [James et al. \(2021, chap. 7.4\)](#) and [Hastie, Tibshirani, and Friedman \(2009, chap. 5.2\)](#). Standard econometrics references for nonparametric and semiparametric regression are [Pagan and Ullah \(1999\)](#) and [Li and Racine \(2007\)](#). Chapter 27 provides much greater coverage of nonparametric and semiparametric regression. Quantile regression, also a semiparametric method, is covered in chapter [15](#).

## 14.9 Exercises

1. Using the code of section [14.2.3](#) as a template and setting sample size at 1,000, generate a well-separated mixture of  $t(3)$ -distributed univariate variables with mixing proportions 0.2, 0.5, and 0.5. Use the `fmm` prefix to estimate the mixture parameters.
2. Modify your code for exercise 1 to generate a two-component mixture of normals with regression components  $(\beta_{11} + \beta_{12}x)$  and  $(\beta_{21} + \beta_{22}x)$  with values of component-specific parameters, regressor  $x$ , and  $\sigma_1^2$  and  $\sigma_2^2$  set at values of your own choosing. Fit the mixture model using the `fmm` prefix. Using the postestimation command `predict`, generate posterior probabilities for each component. Use the `summarize` command to verify that the mean of posterior probabilities is close to the estimated mixture proportions.
3. Using the generated data in question 2, estimate a pooled OLS regression using the `regress` command. How are the estimated parameters related to those for question 2?
4. Continuing from the previous question, use the `regress` postestimation command `predict` with option `residuals` to generate the OLS residuals. Use the `kdensity` command to plot the kernel density of the residuals. Check whether the plot corresponds to the DGP of a well-separated mixture.
5. Consider piecewise linear regression with one knot at  $c$ , with  $f(x) = \beta_0 + \beta_1 x + \beta_2(x - c)_+$ . Find functions  $f_1(x) = a_1 + b_1 x$  such that  $f_1(x) = f(x)$  for  $x < c$  and  $f_2(x) = a_2 + b_2 x$  such that  $f_2(x) = f(x)$  for  $x \geq c$  (so express  $a_1, b_1$  as functions of  $\beta_0, \beta_1, \beta_2$  and similarly for  $a_2, b_2$ ). Then, show  $f_1(c) = f_2(c)$  so the function is continuous at  $c$ .
6. For simplicity, consider a quadratic regression spline, rather than a cubic spline, with one knot. Then  $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_4(x - c)_+^2$ . Show that this formula yields distinct quadratic polynomials for  $x \leq c$  and  $x > c$  with  $f(c-) = f(c)$  and  $f'(c-) = f'(c)$ . First, find  $f_1(x) = a_1 + b_1 x + c_1 x^2$  such that  $f_1(x) = f(x)$  for  $x < c$  and  $f_2(x) = a_2 + b_2 x + c_2 x^2$  such that  $f_2(x) = f(x)$  for  $x \geq c$  (so express  $a_1, b_1, c_1$  as functions of

$\beta_0, \beta_1, \beta_2, \beta_3$  and similarly for  $a_2, b_2, c_2$ ). Then, show  $f_1(c) = f_2(c)$  and  $f'_1(c) = f'_2(c)$ .

7. Use `mus214mabelfmm.dta`. Obtain a graph with a scatterplot of `logyearn` against `logyhrs` and a fitted linear spline where the spline knots are created using command `mkspline` with option `pctile` and the number of knots is set to 5. Does the relationship appear to be linear? Obtain a similar graph for a fitted cubic spline with knots created using command `mkspline` with options `cubic` and `knots 5`. Comment on any difference in the fitted curves. Are the same knots used? Hint: Use option `displayknots` of command `mkspline`.
8. Use `mus214mabelfmm.dta`, and create a subsample using commands `set seed 101` and `keep if runiform() < 0.02`. Provide local linear nonparametric graphs with the `epan2` kernel of `logyearn` against `logyhrs` using the `lpoly` command and using the `npregress kernel` and `npgraph` commands. Does the relationship appear to be linear?
9. Continue with the previous question. Use the `bwidth()` option of the `lpoly` command to set the bandwidth to be equal to that used by the `npregress kernel` command. Do the plots appear to be the same? Are the predictions from `lpoly` and `npregress kernel` identical?



# **Chapter 15**

## **Quantile regression**

## 15.1 Introduction

The standard linear regression is a useful tool for summarizing the average relationship between the outcome variable of interest and a set of regressors, based on the conditional mean regression function  $E(y|\mathbf{x})$ . This provides only a partial view of the relationship. A more complete picture would provide information about the relationship between the outcome  $y$  and the regressors  $\mathbf{x}$  at different points in the conditional distribution of  $y$ . Quantile regression (QR) is a statistical tool for building just such a picture.

Quantiles and percentiles are synonymous—the 0.99 quantile is the 99th percentile. The median, defined as the middle value of a set of ranked data, is the best-known specific quantile. The sample median is an estimator of the population median. If  $F(y) = \Pr(Y \leq y)$  defines the cumulative distribution function (c.d.f.), then  $F(y_{\text{med}}) = 1/2$  is the equation whose solution defines the median  $y_{\text{med}} = F^{-1}(1/2)$ , or more precisely, the infimum of  $F^{-1}(1/2)$  to ensure uniqueness. The quantile  $q$ ,  $q \in (0, 1)$ , is defined as that value of  $y$  that splits the data into the proportions  $q$  below and  $1 - q$  above; that is,  $F(y_q) = q$  and  $y_q = F^{-1}(q)$ . For example, if  $y_{0.99} = 200$ , then  $\Pr(Y \leq 200) = 0.99$ . In practice,  $F(y)$  may have flat sections and jumps, so a more precise statement is that

$$y_q = \inf\{y \in R : q \leq F(y)\}.$$

These concepts extend to the conditional quantile regression (CQR) function, denoted as  $Q_q(y|\mathbf{x})$ , where the function  $Q_q(y|\mathbf{x})$  is usually specified to be linear in parameters.

CQRs have considerable appeal for several reasons. Median regression, also called least-absolute-deviations regression, is a special case of CQR and is more robust to outliers than is mean regression; see section [3.6.5](#). CQR, as we shall see, permits us to study the impact of regressors on both the location and scale parameters of the model, thereby allowing a richer understanding of the data. The approach is semiparametric in the sense that it avoids assumptions about the parametric distribution of regression errors. It is also robust with respect to departures from normality.

QR is increasingly used to study distributional issues, such as the impact of a policy treatment on various quantiles of the distribution of the outcome variable. For example, one may be interested in the effect of training on earnings at various points of the earnings distribution while also controlling for individual characteristics, an example of heterogeneous treatment effects. Furthermore, interest may lie in the impact on unconditional quantiles, in addition to the impact on conditional quantiles, and alternative unconditional QR methods are used. And in many applications, the policy treatment may be endogenous.

This chapter explores the application of CQR using several Stata commands. QR with an endogenous regressor and unconditional QR, more advanced topics that often use the treatment evaluation framework, are presented in chapter 25.

## 15.2 Conditional quantile regression

In this section, we review the theoretical background of CQR analysis.

Let  $e_i$  denote the model prediction error. Then ordinary least squares (OLS) minimizes  $\sum_i e_i^2$ , and median regression minimizes  $\sum_i |e_i|$ . As explained below, conditional QR at the  $q$ th quantile minimizes a sum that gives the asymmetric penalties  $(1 - q)|e_i|$  for overprediction and  $q|e_i|$  for underprediction. Linear programming methods need to be used to obtain the conditional QR estimator, but it is still asymptotically normally distributed and is easily obtained using Stata commands.

### 15.2.1 Conditional quantiles

Many applied econometrics studies model conditional moments, especially the conditional mean function. Suppose that the main objective of modeling is the conditional prediction of  $y$  given  $\mathbf{x}$ . Let  $\hat{y}(\mathbf{x})$  denote the predictor function and  $e(\mathbf{x}) \equiv y - \hat{y}(\mathbf{x})$  denote the prediction error. Then,

$$L\{e(\mathbf{x})\} = L\{y - \hat{y}(\mathbf{x})\}$$

denotes the loss associated with the prediction error  $e$ . The optimal loss-minimizing predictor depends upon the function  $L(\cdot)$ . If  $L(e) = e^2$ , then the conditional mean function,  $E(y|\mathbf{x}) = \mathbf{x}'\beta$  in the linear case, is the optimal predictor. If the loss criterion is absolute error loss, so  $L(e) = |e|$ , then the optimal predictor is the conditional median, denoted by  $\text{med}(y|\mathbf{x})$ . If the conditional median function is linear, so that  $\text{med}(y|\mathbf{x}) = \mathbf{x}'\beta$ , then the optimal predictor is  $\hat{y} = \mathbf{x}'\hat{\beta}$ , where  $\hat{\beta}$  is the least-absolute-deviations estimator that minimizes  $\sum_i |y_i - \mathbf{x}'_i\beta|$ .

Both the squared-error and absolute-error loss functions are symmetric, which implies that the same penalty is imposed for prediction error of a given magnitude regardless of the direction of the prediction error. The asymmetry parameter  $q$  is specified. It lies in the interval  $(0, 1)$  with

symmetry when  $q = 0.5$  and increasing asymmetry as  $q$  approaches 0 or 1. Then the optimal predictor is the  $q$ th conditional quantile, denoted by  $Q_q(y|\mathbf{x})$ , and the conditional median is a special case when  $q = 0.5$ . CQR involves inference regarding the conditional quantile function.

Standard CQR analysis assumes that the conditional QR  $Q_q(y|\mathbf{x})$  is linear in  $\mathbf{x}$ ; for nonparametric QR, see [Koenker \(2005\)](#).

Quite apart from the considerations of loss function (on which agreement may be difficult to obtain), there are several attractive features of CQR. First, unlike the OLS regression, which is sensitive to the presence of outliers and can be inefficient when the dependent variable has a highly nonnormal distribution, the conditional QR estimates are more robust. Second, QR also provides a potentially richer characterization of the data. For example, QR allows us to study the impact of a covariate on the full distribution or any particular percentile of the distribution, not just the conditional mean. Third, unlike OLS, QR estimators do not require existence of the conditional mean for consistency. Finally, conditional QR is equivariant to monotone transformations. This means that the quantiles of a transformed variable  $y$ , denoted by  $h(y)$ , where  $h(\cdot)$  is a monotonic function, equal the transforms of the quantiles of  $y$ , so  $Q_q\{h(y)\} = h\{Q_q(y)\}$ . Hence, if the quantile model is expressed as  $h(y)$ , for example,  $\ln y$ , then one can use the inverse transformation to translate the results back to  $y$ . This is not possible for the mean, because  $E\{h(y)\} \neq h\{E(y)\}$ . The equivariance property for quantiles continues to hold in the regression context, assuming that the conditional quantile model is correctly specified; see section [15.3.4](#).

### 15.2.2 Computation of conditional QR estimates

In the nonregression case, the  $q$ th sample quantile  $y_q = F^{-1}(q)$  can be obtained as the solution to minimizing the objective function

$$\begin{aligned}
S(y_q) &= \sum_{i:y_i \geq y_q}^N q|y_i - y_q| + \sum_{i:y_i < y_q}^N (1-q)|y_i - y_q| \\
&= \sum_{i=1}^N (y_i - y_q)\{q - \mathbf{1}(y_i - y_q > 0)\} \\
&= \sum_{i=1}^N \rho_q(y_i - y_q)
\end{aligned} \tag{15.1}$$

where

$$\rho_q(u) = u\{q - \mathbf{1}(u < 0)\} \tag{15.2}$$

is called the check function. In practice, a range of values solves this, in which case  $y_q$  is the smallest value because a c.d.f. is defined to be left continuous.

For regression, by extension of the first equation in (15.1), the  $q$ th CQR estimator  $\hat{\beta}_q$  minimizes over  $\beta_q$  the objective function

$$S(\beta_q) = \sum_{i:y_i \geq \mathbf{x}'_i \beta}^N q|y_i - \mathbf{x}'_i \beta_q| + \sum_{i:y_i < \mathbf{x}'_i \beta}^N (1-q)|y_i - \mathbf{x}'_i \beta_q| \tag{15.3}$$

where  $0 < q < 1$ , and we use  $\beta_q$  rather than  $\beta$  to make clear that different choices of  $q$  estimate different values of  $\beta$ . If  $q = 0.9$ , for example, then much more weight is placed on prediction for observations with  $y \geq \mathbf{x}' \beta_{0.9}$  than for observations with  $y < \mathbf{x}' \beta_{0.9}$ . Often, estimation sets  $q = 0.5$ , giving the least-absolute-deviations estimator that minimizes  $\sum_i |y_i - \mathbf{x}'_i \beta_{0.5}|$ .

Using the third equation in (15.1), we more simply write the objective function (15.3) as

$$Q(\boldsymbol{\beta}_q) = \sum_{i=1}^N \rho_q(y_i - \mathbf{x}'_i \boldsymbol{\beta})$$

where  $\rho_q(u)$  is the check function defined in (15.2). The objective function is not differentiable, so the usual gradient optimization methods cannot be applied. Instead, it is a linear program. The classic solution method is the simplex method, which is guaranteed to yield a solution in a finite number of simplex iterations.

### 15.2.3 Computation of CQR standard errors

The estimator that minimizes  $Q(\boldsymbol{\beta}_q)$  is an  $m$  estimator with well-established asymptotic properties. The CQR estimator is asymptotically normal under general conditions; see [Cameron and Trivedi \(2005, 88\)](#).

Under the assumption of independent heteroskedastic errors, it can be shown that

$$\widehat{\boldsymbol{\beta}}_q \stackrel{a}{\sim} N(\boldsymbol{\beta}_q, \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-1}) \quad (15.4)$$

where  $\mathbf{A} = E\{f_{u_q}(0|\mathbf{x}_i)\mathbf{x}_i\mathbf{x}'_i\}$ ,  $\mathbf{B} = E\{q(1-q)\mathbf{x}_i\mathbf{x}'_i\}$ , and  $f_{u_q}(0|\mathbf{x}_i)$  is the conditional density of the error term  $u_{qi} = y_i - \mathbf{x}'_i \boldsymbol{\beta}_q$  evaluated at  $u_{qi} = 0$ . This analytical expression involves  $f_{u_q}(0|\mathbf{x}_i)$ , a quantity that can be estimated in several ways. The Stata default uses a function of the fitted values; an alternative method uses a kernel-density estimate. Other methods for obtaining heteroskedastic–robust standard errors are to use a paired bootstrap (see sections [3.4.7](#) and [12.3.1](#)) or to use White heteroskedastic–robust standard errors.

The result (15.4) has been extended to the clustered case by [Parente and Santos Silva \(2016\)](#). Alternatively, one can use a cluster pairs bootstrap (see

section [12.3.5](#)). With few clusters, a better method uses a wild gradient bootstrap proposed by [Hagemann \(2020\)](#).

### 15.2.4 The `qreg`, `bsqreg`, `sqreg`, and `qreg2` commands

The Stata commands for CQR estimation are similar to those for ordinary regression. There are three variants—`qreg`, `bsqreg`, and `sqreg`—that are commonly used. The first two are used for estimating a CQR for a specified value of  $q$ , without or with bootstrap standard errors, respectively. The `sqreg` command is used when several different values of  $q$  are specified simultaneously. A fourth command, used less frequently, is `iqreg`, for interquantile regression.

The basic QR command is `qreg`, with the following syntax:

```
qreg depvar [indepvars] [if] [in] [weight] [, options]
```

A simple example with the `qreg` options set to default is `qreg y x z`. This will estimate the median regression,  $y_{\text{med}} = \beta_1 + \beta_2 x + \beta_3 z$ ; that is, the default  $q$  is 0.5. The `quantile()` option allows one to choose  $q$ . For example, `qreg y x z, quantile(.75)` sets  $q = 0.75$ . Other options include `level(#)` to set the level for reported confidence intervals and an optimization-related option.

The default standard errors assume independent and identically distributed (i.i.d.) errors, in which case the analytical formula ([15.4](#)) reduces to  $\mathbf{A}^{-1}$ , where  $\mathbf{A} = \sum_i q(1 - q)\mathbf{x}_i\mathbf{x}'_i/f_{u_q}^2(0)$ . Various options can be used to change the method used to compute  $f_{u_q}(0)$ . The `vce(robust)` option for `qreg` obtains White heteroskedastic–robust standard errors.

`bsqreg` obtains heteroskedastic–robust standard errors by instead using a pairs bootstrap. The standard errors from `bsqreg` are heteroskedastic robust in the same sense as those from `vce(robust)` for other commands. The command syntax is the same as for `qreg`. A key option is `reps(#)`, which sets the number of bootstrap replications. This option should be used because the default is only 20. And for replicability of results, one should

first issue the `set seed` command. For example, give the commands `set seed 10101` and `bsqreg y x z, reps(400) quantile(.75)`.

The `iqreg` command for interquantile regression reports the difference in coefficient estimates from separate CQR estimation at two different values of  $q$ .

When QR estimates are obtained for several values of  $q$ , and we want to test whether regression coefficients for different values of  $q$  differ, the `sqreg` command is used. This provides coefficient estimates and an estimate of the simultaneous or joint variance–covariance matrix of the estimator of  $\hat{\beta}_q$  across different specified values of  $q$ , using the bootstrap. The command syntax is again the same as `qreg` and `bsqreg`, and several quantiles can now be specified in the `quantile()` option. For example, `sqreg y x z, quantile(.2 .5 .8) reps(400)` produces QR estimates for  $q = 0.2$ ,  $q = 0.5$ , and  $q = 0.8$ , together with bootstrap standard errors based on 400 replications.

The `qreg2` command by [Machado, Parente, and Santos Silva \(2011\)](#) obtains heteroskedastic-robust standard errors using the analytical formula ([15.4](#)).

The `cluster()` option of the `qreg2` command computes the cluster extension of formula ([15.4](#)) due to [Parente and Santos Silva \(2016\)](#). Alternatively, cluster-robust standard errors can be obtained by manually implementing a bootstrap by applying the `bootstrap` prefix, with option `cluster()`, to the `qreg` estimation command.

## 15.3 CQR for medical expenditures data

We present the basic QR commands applied to the log of medical expenditures.

### 15.3.1 Data summary

The data used in this example come from the Medical Expenditure Panel Survey and are identical to those discussed in section [3.2](#). Again, we consider a regression model of total medical expenditure by the Medicare elderly. The dependent variable is `ltotexp`, so observations with zero expenditures are omitted. The explanatory variables are an indicator for supplementary private insurance (`suppins`), one health-status variable (`totchr`), and three sociodemographic variables (`age`, `female`, `white`).

We first summarize the data:

```
. * Read in log of medical expenditures data and summarize  
. qui use mus203mepsmedexp  
. drop if ltotexp == .  
(109 observations deleted)  
. summarize ltotexp suppins totchr age female white, separator(0)
```

Variable	Obs	Mean	Std. dev.	Min	Max
ltotexp	2,955	8.059866	1.367592	1.098612	11.74094
suppins	2,955	.5915398	.4916322	0	1
totchr	2,955	1.808799	1.294613	0	7
age	2,955	74.24535	6.375975	65	90
female	2,955	.5840948	.4929608	0	1
white	2,955	.9736041	.1603368	0	1

The major quantiles of `ltotexp` can be obtained by using `summarize`, `detail`, and specific quantiles can be obtained by using the `centile` command. The same estimates for a specific quantile can also be obtained by QR on an intercept only. We have

```
. * Intercept-only quantile regression gives the raw quantile
. centile ltotexp, centile(25)
```

Variable	Obs	Percentile	Centile	Binom. interp.	
				[95% conf. interval]	
ltotexp	2,955	25	7.267525	7.200001	7.339215

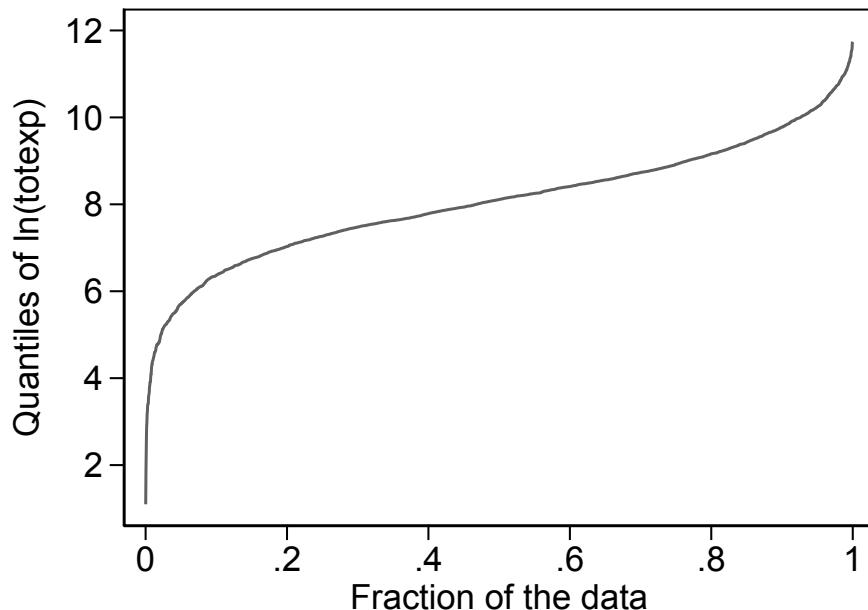
```
. qreg ltotexp, quantile(.25) nolog
.25 Quantile regression
Raw sum of deviations 1293.577 (about 7.2675252)
Min sum of deviations 1293.577
```

Variable	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
_cons	7.267525	.0341235	212.98	0.000	7.200617	7.334433

The community-contributed `qplot` command due to [Cox \(2005\)](#) is a useful visual aid because it provides a plot of all quantiles. We have

```
. * Quantile plot for ltotexp using the community-contributed command qplot
. qplot ltotexp, recast(line) scale(1.5) ytitle("Quantiles of ln(totexp)")
> xtitle("Fraction of the data")
```

The plot, shown in figure 15.1, is the same as a plot of the empirical c.d.f. of `ltotexp`, except that the axes are reversed. We have, very approximately,  $q_{0.1} = 6$ ,  $q_{0.25} = 7$ ,  $q_{0.5} = 8$ ,  $q_{0.75} = 9$ , and  $q_{0.9} = 10$ . The distribution appears to be reasonably symmetric, at least for  $0.05 < q < 0.95$ .



**Figure 15.1.** Quantiles of the dependent variable

### 15.3.2 Conditional QR estimates

The basic CQR output for the median regression, with standard errors computed using the default option, is obtained using the `qreg` command with  $q = 0.5$ .

```
. * Basic quantile regression for q = 0.5
. qreg ltotexp suppins totchr age female white
Iteration 1: WLS sum of weighted deviations = 1400.8169
Iteration 1: sum of abs. weighted deviations = 1400.9985
Iteration 2: sum of abs. weighted deviations = 1399.797
Iteration 3: sum of abs. weighted deviations = 1399.7529
Iteration 4: sum of abs. weighted deviations = 1399.5861
Iteration 5: sum of abs. weighted deviations = 1398.9092
Iteration 6: sum of abs. weighted deviations = 1398.8274
note: alternate solutions exist.
Iteration 7: sum of abs. weighted deviations = 1398.5229
Iteration 8: sum of abs. weighted deviations = 1398.522
Iteration 9: sum of abs. weighted deviations = 1398.5155
Iteration 10: sum of abs. weighted deviations = 1398.5105
Iteration 11: sum of abs. weighted deviations = 1398.5067
Iteration 12: sum of abs. weighted deviations = 1398.4992
Iteration 13: sum of abs. weighted deviations = 1398.498
Iteration 14: sum of abs. weighted deviations = 1398.4951
Iteration 15: sum of abs. weighted deviations = 1398.4945
Iteration 16: sum of abs. weighted deviations = 1398.4916
Median regression                                         Number of obs =      2,955
  Raw sum of deviations   1555.48 (about 8.111928)
  Min sum of deviations  1398.492                           Pseudo R2      =     0.1009
```

	Coefficient	Std. err.	t	P> t	[95% conf. interval]
ltotexp					
suppins	.2769771	.0535936	5.17	0.000	.1718924 .3820617
totchr	.3942664	.0202472	19.47	0.000	.3545663 .4339664
age	.0148666	.0041479	3.58	0.000	.0067335 .0229996
female	-.0880967	.0532006	-1.66	0.098	-.1924109 .0162175
white	.4987457	.1630984	3.06	0.002	.1789474 .818544
_cons	5.648891	.341166	16.56	0.000	4.979943 6.317838

The iterations here refer to simplex iterations rather than the usual Newton–Raphson (or related gradient-method) iterations. All regressors, aside from `female`, are highly statistically significant with expected signs.

### 15.3.3 Interpretation of conditional quantile coefficients

The key advantage of CQR is that it can accommodate heterogeneous treatment effects, whereby a change in a specific regressor can have different effects at different quantiles of the conditional distribution of the outcome of interest.

For example, one more year of schooling may have a larger effect for someone at a low conditional quantile of earnings than for someone with high predicted earnings given his or her various measured characteristics.

Such heterogeneous effects are likely even if the conditional quantile function is linear. To see this, consider a single regressor, for simplicity, with linear model

$$y_i = \beta_1 + \beta_2 x_i + u_i \quad (15.5)$$

where the error  $u_i$  satisfies  $E(y_i|x_i) = 0$ . We denote the  $q$ th conditional quantile function of  $y$  given  $x$  as  $Q_q(y|x)$ , analogous to using  $E(y|x)$  to denote the conditional mean. Conditioning on  $x_i$ , (15.5) implies that

$$Q_q(y_i|x_i) = \beta_1 + \beta_2 x_i + F_{u_i|x_i}^{-1}(q)$$

where  $F_{u_i|x_i}$  is the conditional distribution function of  $u_i|x_i$ . Conditional on  $x_i$ , the quantile depends on the distribution of  $u_i|x_i$  via the term  $F_{u_i|x_i}^{-1}(q)$ . This will depend on  $x_i$  if, for example, errors are heteroskedastic. Then, in general,  $Q_q(y|x)$  at different values of  $q$  will differ in more than just the intercept and may well even be nonlinear in  $x$ .

When are the quantile effects homogeneous? In the special case that errors are i.i.d., considerable simplification occurs as  $F_{u_i|x_i}^{-1}(q) = F_u^{-1}(q)$ , which does not vary with  $i$ . Then the conditional quantile is

$$Q_q(y_i|x_i) = \{\beta_1 + F_u^{-1}(q)\} + \beta_2 x_i$$

Here the conditional quantile functions at different quantiles have a common slope, differing only in the intercepts  $\beta_0 + F_u^{-1}(q)$ . In such a simple case, there is no need to use CQR to obtain marginal effects (MES) at different quantiles because the quantile slope coefficient  $\beta_2$  does not vary with the quantile.

More generally, we condition on many individual characteristics. Thus, we may be interested in the heterogeneous effect of schooling at different conditional quantiles where conditioning is on many observed characteristics and not just on schooling.

In the multiple regression model, the standard linear conditional quantile function is

$$Q_q(y_i|\mathbf{x}_i) = \mathbf{x}'_i \boldsymbol{\beta}_q$$

The MES after CQR can be obtained in the usual way. For the  $j$ th (continuous) regressor, the ME is

$$\frac{\partial Q_q(y|\mathbf{x})}{\partial x_j} = \beta_{qj}$$

and in the discrete case is  $Q_q(y|\mathbf{x} + \Delta x_j) - Q_q(y|\mathbf{x}) = \beta_{qj} \times \Delta x_j$ . As for linear least-squares regression, the ME at a given conditional quantile is given by the slope coefficient and is invariant across individuals, simplifying analysis. However, the interpretation is somewhat delicate for discrete changes that are more than infinitesimal because the partial derivative measures the impact of a change in  $x_j$  under the assumption that the individual remains in the same conditional quantile of the distribution after the change. For larger changes in a regressor, the individual may shift into a different conditional quantile.

The preceding estimates are that the conditional median of `ltotexp`, where conditioning is on `suppins`, `totchr`, `age`, `female`, and `white`, increases by 0.277 for an individual with supplementary insurance, increases by 0.394 with each additional chronic condition, increases by 0.015 with each year of aging, and so on.

The Stata QR commands are for models that are linear in parameters but not necessarily linear in variables. For example, we could include as regressors a quadratic in age as `c.age##c.age`, using factor notation, and obtain the ME of age after CQR using the postestimation command `margins, dydx(*)`.

### 15.3.4 Retransformation

For our example, with the dependent variable  $\text{ltotexp} = \ln(\text{totexp})$ , the results from `qreg` give MES for  $\ln(\text{totexp})$ . We may want instead to compute the ME on `totexp`, not `ltotexp`.

The equivariance property of QR is relevant. Given  $Q_q(\ln y|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}_q$ , we have  $Q_q(y|\mathbf{x}) = \exp\{Q_q(\ln y|\mathbf{x})\} = \exp(\mathbf{x}'\boldsymbol{\beta}_q)$ . The ME on  $y$  in levels, given QR model  $\mathbf{x}'\boldsymbol{\beta}_q$  in logs, is then

$$\frac{\partial Q_q(y|\mathbf{x})}{\partial x_j} = \exp(\mathbf{x}'\boldsymbol{\beta}_q)\beta_{qj}$$

This depends on  $\mathbf{x}$ . The average marginal effect (AME) is  $\{N^{-1} \sum_{i=1}^N \exp(\mathbf{x}_i'\boldsymbol{\beta}_q)\}\beta_{qj}$  and can be estimated if we use the `predict` postestimation command to obtain  $\exp(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_q)$  and then average. We obtain

```
. * Obtain multiplier to convert QR coeffs in logs to AME in levels.
. qui predict xb
. generate expxb = exp(xb)
. qui summarize expxb
. display "Multiplier of QR in logs coeffs to get AME in levels = " r(mean)
Multiplier of QR in logs coeffs to get AME in levels = 3746.7178
```

For example, the AME of `totchr` on  $\ln(\text{totexp})$  is 0.3943 from the earlier `qreg` output above. The implied AME of `totchr` on the levels variable is therefore  $3746.7 \times 0.3943 = 1477$ . One more chronic condition increases the conditional median of expenditures by \$1,477.

The equivariance property that  $Q_q(y|\mathbf{x}) = \exp\{Q_q(\ln y|\mathbf{x})\}$  is exact only if the conditional quantile function is correctly specified. This is unlikely to be the case because the linear model will inevitably be only an approximation. One case where the linear model is exact is where all regressors are discrete and we specify a fully saturated model with indicator variables as regressors that exhaust all possible interactions between discrete regressors. We pursue this in the second end-of-chapter exercise.

### 15.3.5 Comparison of estimates at different quantiles

CQRs are usually performed at different quantiles. Here we do so for the quartiles  $q = 0.25$ ,  $0.50$ , and  $0.75$  and compare the results with one another and with OLS estimates. White heteroskedastic–robust standard errors are reported. We obtain

```

. * Compare (1) OLS; (2-4) coeffs across quantiles .25, .50, and .75
. qui regress ltotexp suppins totchr age female white, vce(robust)
. estimates store OLS
. qui qreg ltotexp suppins totchr age female white, quantile(.25) vce(robust)
. estimates store QR_25
. qui qreg ltotexp suppins totchr age female white, quantile(.50) vce(robust)
. estimates store QR_50
. qui qreg ltotexp suppins totchr age female white, quantile(.75) vce(robust)
. estimates store QR_75
. estimates table OLS QR_25 QR_50 QR_75, b(%7.3f) se

```

Variable	OLS	QR_25	QR_50	QR_75
suppins	0.257	0.386	0.277	0.149
	0.047	0.060	0.053	0.062
totchr	0.445	0.459	0.394	0.374
	0.017	0.018	0.018	0.023
age	0.013	0.016	0.015	0.018
	0.004	0.004	0.004	0.005
female	-0.077	-0.016	-0.088	-0.122
	0.046	0.053	0.054	0.061
white	0.318	0.338	0.499	0.193
	0.136	0.097	0.193	0.257
_cons	5.898	4.748	5.649	6.600
	0.294	0.307	0.352	0.427

Legend: b/se

OLS coefficients differ considerably from the CQR coefficients, even those for median regression.

The CQR coefficients vary across quantiles. Most noticeably, the highly statistically significant regressor `suppins` (supplementary insurance) has a much greater impact at the lower conditional quantile. Note that this implies that `suppins` has a larger effect for those with log expenditure considerably less than that predicted given their individual characteristics. It does not necessarily imply that `suppins` has a larger effect for those with lower log expenditures. For example, the lower conditional quartile might actually be composed of people with high expenditures.

The standard errors are smaller for median regression ( $q = 0.50$ ) than for the upper and lower quantiles ( $q = 0.25, 0.75$ ), reflecting more precision at the center of the distribution.

### 15.3.6 Robust variance estimation using the qreg2 command

The community-contributed `qreg2` command by [Machado, Parente, and Santos Silva \(2011\)](#) obtains heteroskedastic–robust standard errors using a consistent estimator of the analytical formula (15.4) and also obtains cluster–robust standard errors using the cluster extension of formula (15.4) due to [Parente and Santos Silva \(2016\)](#). It additionally provides a formal test of heteroskedasticity.

The command has a format similar to the `qreg` command. The following provides heteroskedastic–robust standard errors following median regression.

```
. * qreg2 provides heteroskedastic-robust standard errors and also tests for
> heteroskedasticity
. qreg2 ltotexp suppins totchr age female white, quantile(0.5)
Median regression
R-squared = .1954566
Number of obs = 2955
Objective function = .47326279
```

Heteroskedasticity robust standard errors

ltotexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
suppins	.2769771	.0541489	5.12	0.000	.1708037 .3831505
totchr	.3942664	.0202004	19.52	0.000	.354658 .4338748
age	.0148666	.0040878	3.64	0.000	.0068513 .0228819
female	-.0880967	.0532428	-1.65	0.098	-.1924936 .0163002
white	.4987457	.194073	2.57	0.010	.1182135 .8792779
_cons	5.648891	.3637387	15.53	0.000	4.935683 6.362098

```
Machado-Santos Silva test for heteroskedasticity
Ho: Constant variance
Variables: Fitted values of ltotexp and its squares
chi2(2)      =  56.274
Prob > chi2  =  0.000
```

The point estimates are identical to those from `qreg` in section [15.3.2](#), and the robust standard errors are very similar.

The output includes a test for heteroskedasticity that strongly rejects the null hypothesis of homoskedastic model errors, so that inference needs to be based on the computationally more intensive heteroskedastic–robust

standard errors. The test is against heteroskedasticity of the form  $\text{Var}(u|\mathbf{x}) = h(\alpha_1 + \mathbf{z}'\boldsymbol{\alpha}_2)$ , where the command default is to set  $\mathbf{z}_i = (\hat{y}_{qi}, \hat{y}_{qi}^2)$ , where  $\hat{y}_{qi} = \mathbf{x}'_i \hat{\boldsymbol{\beta}}_q$ .

The `cluster()` option of the `qreg2` command provides cluster-robust standard errors and provides a formal test of the null hypothesis of intracluster correlation.

### 15.3.7 Comparison of different standard-error estimates

There are several different methods for computing QR standard errors. In general, it is best to use heteroskedastic-robust standard errors with independent data and cluster-robust standard errors with clustered data (with many clusters). The analytical methods to obtain these robust standard errors involve complications such as kernel-density estimation at zero, while bootstrap methods are more computationally intensive.

The following median regression example presents default `qreg` standard errors that assume i.i.d. errors, followed by three different methods for computing heteroskedastic-robust standard errors, followed by two methods for computing cluster-robust standard errors where the clustering is on `educ`.

```

. * Median regression: i.i.d., heteroskedastic-robust, and cluster--robust
> standard errors
. qui qreg ltotexp suppins totchr age female white, quantile(.50)
. estimates store IID
. qui qreg ltotexp suppins totchr age female white, quantile(.50) vce(robust)
. estimates store HETWHITE
. qui qreg2 ltotexp suppins totchr age female white, quantile(.50)
. estimates store HETANAL
. set seed 10101
. qui bsqreg ltotexp suppins totchr age female white, quant(.50) reps(400)
. estimates store HETBOOT
. qui qreg2 ltotexp suppins totchr age female white, quant(.50) cluster(educyr)
. estimates store CLUANAL
. qui bootstrap, cluster(educyr) seed(10101) reps(400):
>     qreg ltotexp suppins totchr age female white, quant(.50)
. estimates store CLUBOOT
. estimates table IID HETWHITE HETANAL HETBOOT CLUANAL CLUBOOT, b(%7.3f) se

```

Variable	IID	HETWH~E	HETANAL	HETBOOT	CLUANAL	CLUBOOT
suppins	0.277 0.054	0.277 0.053	0.277 0.054	0.277 0.057	0.277 0.066	0.277 0.073
totchr	0.394 0.020	0.394 0.018	0.394 0.020	0.394 0.019	0.394 0.015	0.394 0.018
age	0.015 0.004	0.015 0.004	0.015 0.004	0.015 0.004	0.015 0.004	0.015 0.004
female	-0.088 0.053	-0.088 0.054	-0.088 0.053	-0.088 0.052	-0.088 0.051	-0.088 0.063
white	0.499 0.163	0.499 0.193	0.499 0.194	0.499 0.199	0.499 0.181	0.499 0.174
_cons	5.649 0.341	5.649 0.352	5.649 0.364	5.649 0.391	5.649 0.306	5.649 0.369

Legend: b/se

In this example, there is at most 30% variation across the various standard errors. Because we use the log of medical expenditures, heteroskedasticity in a linear regression is expected to be greatly reduced, though this does not necessarily imply that heteroskedasticity in the errors from median regression is reduced. The cluster-robust standard errors were clustered on `educ` for illustration and were not necessarily expected to make a big difference compared with heteroskedastic-robust standard errors.

### 15.3.8 Heteroskedasticity test based on OLS regression

One reason for coefficients differing across quantiles is the presence of heteroskedastic errors. From OLS output not shown here, the OLS standard errors are similar whether the default or robust estimates are obtained, suggesting little heteroskedasticity. And the logarithmic transformation of the dependent variable that has been used often reduces heteroskedasticity.

We use `estat hettest` to test against heteroskedasticity, which depends on the same variables as those in the OLS regression. Then,

```
. * Test for heteroskedasticity in linear model using estat hettest
. qui regress ltotexp suppins totchr age female white
. estat hettest suppins totchr age female white, iid
Breusch-Pagan/Cook-Weisberg test for heteroskedasticity
Assumption: i.i.d. error terms
Variables: suppins totchr age female white
H0: Constant variance
chi2(5) = 71.38
Prob > chi2 = 0.0000
```

The null hypothesis of homoskedasticity of errors in OLS regression with dependent variable `ltotexp` is soundly rejected. This is consistent with the earlier finding using the `qreg2` command that homoskedasticity in median regression was rejected.

### 15.3.9 Test of coefficient equality across quantiles

One can conduct hypothesis tests of equality of the regression coefficients at different conditional quantiles.

Consider a test of the equality of the coefficient of `suppins` from QR with  $q = 0.25$ ,  $q = 0.50$ , and  $q = 0.75$ . We first estimate using `sqreg`, rather than `qreg` or `bsqreg`, to obtain the full covariance matrix of coefficients, and then we test. Because this uses the bootstrap, we need to set the seed and number of bootstrap replications.

```

. * Simultaneous QR regression with several values of q
. set seed 10101
. sqreg ltotexp suppins totchr age female white, q(.25 .50 .75) reps(400) nodots
Simultaneous quantile regression                                         Number of obs =      2,955
bootstrap(400) SEs                                                 .25 Pseudo R2 =     0.1292
                                                               .50 Pseudo R2 =     0.1009
                                                               .75 Pseudo R2 =     0.0873

```

ltotexp	Coefficient	Bootstrap			
		std. err.	t	P> t	[95% conf. interval]
q25	suppins	.3856797	.0642742	6.00	0.000
	totchr	.459022	.0234579	19.57	0.000
	age	.0155106	.0043944	3.53	0.000
	female	-.0160694	.0581328	-0.28	0.782
	white	.3375936	.1110348	3.04	0.002
	_cons	4.747962	.3485751	13.62	0.000
q50	suppins	.2769771	.0579685	4.78	0.000
	totchr	.3942664	.0195859	20.13	0.000
	age	.0148666	.0044102	3.37	0.001
	female	-.0880967	.0554863	-1.59	0.112
	white	.4987457	.2199888	2.27	0.023
	_cons	5.648891	.3966791	14.24	0.000
q75	suppins	.1488548	.0649951	2.29	0.022
	totchr	.3735364	.0228424	16.35	0.000
	age	.0182506	.0049533	3.68	0.000
	female	-.1219365	.0562735	-2.17	0.030
	white	.1931923	.2045296	0.94	0.345
	_cons	6.599972	.4247018	15.54	0.000

`sqreg` estimates a QR function for each specified quantile. The coefficients of most of the regressors differ substantially across the quantiles.

We use the `test` command to perform a Wald test of the hypothesis that the coefficients on `suppins` are the same for the three quantiles. Because we are comparing estimates from different equations, we need a prefix to indicate each equation. Here the prefix for the model with  $q = 0.25$ , for example, is `[q25]`. To test that coefficients on the same variable have the same value in different equations, we use the syntax

```
test [ eqname = eqname [= ...] ] [: coeflist]
```

We obtain

```
. * Test of coefficient equality across QR with different q
. test [q25=q50=q75]: suppins
( 1) [q25]suppins - [q50]suppins = 0
( 2) [q25]suppins - [q75]suppins = 0
      F(  2,  2949) =      5.28
      Prob > F =      0.0051
```

The null hypothesis of coefficient equality is rejected at significance level 0.05.

### 15.3.10 Graphical display of coefficients over quantiles

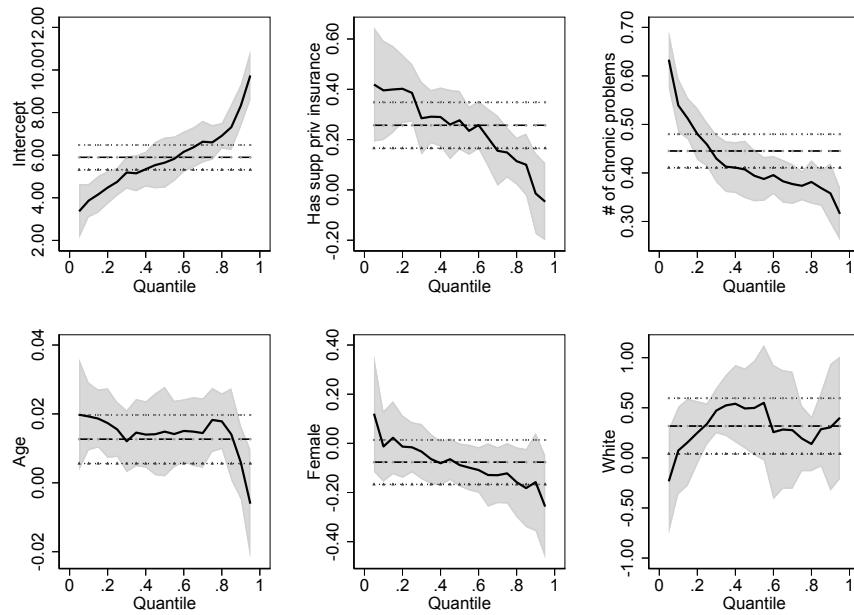
An attractive way to present QR results is via a graphical display of coefficients of interest and their respective confidence intervals. This can be done manually by estimating the parameters of the QR model for a range of values of  $q$ , saving the results to file, and producing separate graphs for each regressor of the estimated coefficient plotted against the quantile  $q$ .

This is done by the community-contributed `grqreg` command (Azevedo 2004), which provides 95% confidence intervals in addition to estimated coefficients. One of the `qreg`, `bsqreg`, or `sqreg` commands must first be executed, and the confidence intervals use the standard errors from whichever command is used. The `grqreg` command does not have enormous flexibility. In particular, it plots coefficients for all regressors, not just selected regressors.

We use `grqreg` with the options `cons` to include the intercept in the graph, `ci` to include a 95% confidence interval, and `ols` and `olsci` to include the OLS coefficient and its 95% confidence interval. The graph option `scale(1.1)` is added to increase the size of the axis titles. The command uses variable labels on the  $y$  axis of each plot, so we provide better variable labels for two of the regressors. We have

```
. * Plots of each regressor's coefficients as quantile q varies
. set seed 10101
. qui bsqreg ltotexp suppins totchr age female white, reps(400)
. grqreg, cons ci ols olsci scale(1.1) seed(10101)
```

In figure 15.2, the horizontal lines are the OLS point estimates and 95% confidence intervals that do not vary with the quantile. The top middle plot shows that the coefficient on `suppins` is positive over most of the range of  $q$ , with a much larger effect at lower conditional quantiles. In the lower conditional quantiles, the point estimates suggest that supplementary insurance is associated with 40% higher medical expenditures (recall that because the dependent variable is in logs, coefficients can be interpreted as semielasticities). Notice that confidence intervals widen at both the extreme upper and lower quantiles.



**Figure 15.2.** QR and OLS coefficients and confidence intervals for each regressor as  $q$  varies from 0 to 1

### 15.3.11 Censored conditional QR

CQR methods are motivated by distributional analysis, but the underlying data are often censored, often from below at zero, as is the case with individual expenditure data. Then we observe  $y_i = \max(y_i^*, c)$ , where  $c$  is the censoring point, and interest lies in the quantiles  $Q_q(y^* | \mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}_q$ .

The community-contributed `cqiv` command implements CQR methods for censored data proposed by [Chernozhukov et al. \(2019\)](#). It additionally allows

for one of the continuous regressors to be endogenous.

## 15.4 CQR for generated heteroskedastic data

To gain more insight on CQR, we consider a simulation example where the quantiles are known to be linear, and we specify a particular form of multiplicative heteroskedasticity using generated data.

### 15.4.1 Simulated dataset

We use a simulated dataset, one where the conditional mean of  $y$  depends on the regressors  $x_2$  and  $x_3$ , while the conditional variance depends on only  $x_2$ .

If  $y = \mathbf{x}'\beta + u$  and  $u = \mathbf{x}'\alpha \times \varepsilon$ , and it is assumed that  $\mathbf{x}'\alpha > 0$  and that  $\varepsilon_i$  is i.i.d., then the quantiles are linear in  $\mathbf{x}$  with the  $q$ th conditional quantile  $Q_q(y|\mathbf{x}) = \mathbf{x}'\{\beta + \alpha \times F_\varepsilon^{-1}(q)\}$ ; see section [15.3.3](#). So for regressors that appear in the conditional mean but not in the heteroskedasticity function (that is,  $\alpha_j = 0$ ), the CQR coefficients do not change with  $q$ , while for other regressors, the coefficients change even though the conditional quantile function is linear in  $\mathbf{x}$ .

If we let  $y = \beta_1 + \beta_2 x_2 + \beta_3 x_3 + u$ , where  $u = (\alpha_1 + \alpha_2 x_2) \times \varepsilon$ , then the CQR coefficients for  $x_2$  will change with  $q$ , while those for  $x_3$  will not. This result requires that  $\alpha_1 + \alpha_2 x_2 > 0$ , so we set  $\alpha_1 > 0$  and  $\alpha_2 > 0$  and generate  $x_2$  from a  $\chi^2(1)$  distribution.

The specific data-generating process (DGP) is

$$\begin{aligned} y &= 1 + 1 \times x_2 + 1 \times x_3 + u; \quad x_2 \sim \chi^2(1), \quad x_3 \sim N(0, 25) \\ u &= (0.1 + 0.5 \times x_2) \times \varepsilon; \quad \varepsilon \sim N(0, 25) \end{aligned}$$

We expect that the CQR estimates of the coefficient of  $x_3$  will take the relatively unchanged value of 1 as the quantiles vary, while the CQR estimates of the coefficient of  $x_2$  will increase as  $q$  increases (because the heteroskedasticity is increasing in  $x_2$ ).

We first generate the data as follows:

```

. * Generated dataset with heteroskedastic errors
. clear all
. set seed 10101
. qui set obs 10000
. generate x2 = rchi2(1)
. generate x3 = 5*rnorm(0)
. generate e = 5*rnorm(0)
. generate u = (.1+0.5*x2)*e
. generate y = 1 + 1*x2 + 1*x3 + u
. summarize e x2 x3 u y

```

Variable	Obs	Mean	Std. dev.	Min	Max
e	10,000	-.0704935	4.978031	-18.59243	19.77932
x2	10,000	.9866486	1.384568	1.95e-09	14.02999
x3	10,000	-.020573	5.018672	-18.65835	16.77711
u	10,000	-.0389596	4.348951	-52.94281	53.22532
y	10,000	1.927116	6.778566	-42.21168	60.20479

The summary statistics confirm that  $x_3$  and  $e$  have approximately a mean of 0 and a variance of 25 and that  $x_2$  has a mean of 1 and a variance of 2, as desired. The output also shows that the heteroskedasticity has induced unusually extreme values of  $u$  and  $y$  that are more than 10 standard deviations from the mean.

We study the distribution of  $y$  further by using several plots. We have

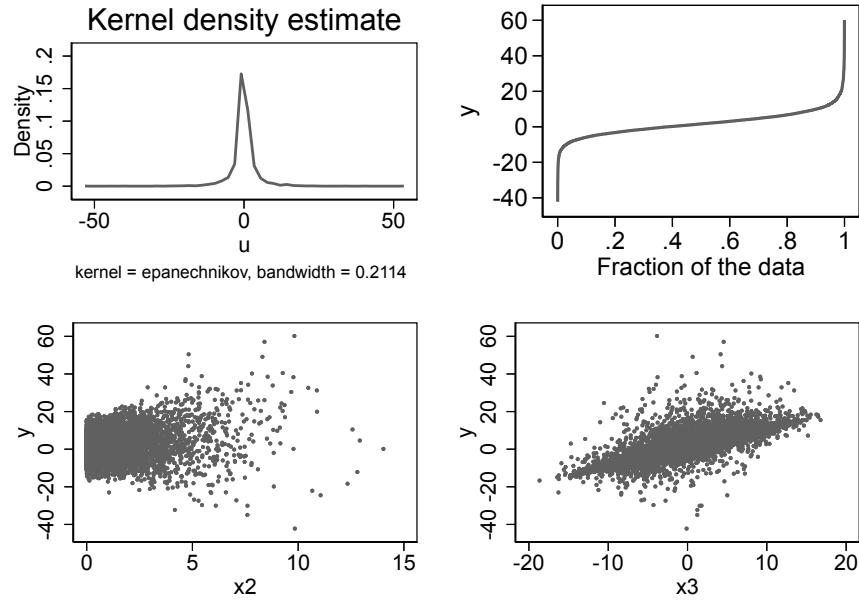
```

. * Generate scatterplots and qplot
. qui kdensity u, scale(1.25) lwidth(medthick) saving(graph1.gph, replace)
. qui qplot y, recast(line) scale(1.4) lwidth(medthick)
> saving(graph2.gph, replace) xtitle("Fraction of the data")
. qui scatter y x2, msize(tiny) scale(1.25) saving(graph3.gph, replace)
. qui scatter y x3, msize(tiny) scale(1.25) saving(graph4.gph, replace)
. graph combine graph1.gph graph2.gph graph3.gph graph4.gph

```

This leads to figure 15.3. The first panel, with the kernel density estimate of  $u$ , shows that the distribution of the error  $u$  is essentially symmetric but has very long tails. The second panel shows the quantiles of  $y$  and indicates symmetry. The third panel plots  $y$  against  $x_2$  and indicates heteroskedasticity and the strongly nonlinear way in which  $x_2$  enters the conditional variance

function of  $y$ . The fourth panel shows no such relationship between  $y$  and  $x_3$



**Figure 15.3.** Density of  $u$ , quantiles of  $y$ , and scatterplots of  $(y, x_2)$  and  $(y, x_3)$

Here  $x_2$  affects both the conditional mean and variance of  $y$ , whereas  $x_3$  enters only the conditional mean function. The regressor  $x_2$  will impact the conditional quantiles differently, whereas  $x_3$  will do so constantly. The OLS regression can display the relationship only between average  $y$  and  $(x_2, x_3)$ . CQR, however, can show the relationship between the regressors and the distribution of  $y$ .

#### 15.4.2 CQR estimates

We next estimate the regression using OLS (with heteroskedasticity-robust standard errors) and QR at  $q = 0.25, 0.50$ , and  $0.75$ , with bootstrap standard errors. The saved results are displayed in a table. The relevant commands and the resulting output are as follows:

```

. * OLS and quantile regression for q = .25, .5, .75
. qui regress y x2 x3
. estimates store OLS
. qui regress y x2 x3, vce(robust)
. estimates store OLS_Rob
. set seed 10101
. qui bsqreg y x2 x3, quantile(.25) reps(400)
. estimates store QR_25
. qui bsqreg y x2 x3, quantile(.50) reps(400)
. estimates store QR_50
. qui bsqreg y x2 x3, quantile(.75) reps(400)
. estimates store QR_75
. estimates table OLS OLS_Rob QR_25 QR_50 QR_75, b(%7.3f) se

```

Variable	OLS	OLS_Rob	QR_25	QR_50	QR_75
x2	0.978 0.031	0.978 0.097	-0.666 0.066	0.986 0.068	2.573 0.077
x3	1.000 0.009	1.000 0.008	1.000 0.003	1.001 0.003	1.003 0.003
_cons	0.983 0.053	0.983 0.068	0.638 0.018	0.977 0.019	1.345 0.019

Legend: b/se

The median regression parameter point estimates of  $\beta_{0.5,2}$  and  $\beta_{0.5,3}$  in this example with symmetrically distributed errors are close to the true values of 1. Interestingly, the median regression parameter estimates are much more precise than the OLS parameter estimates. This improvement is possible because OLS is no longer fully efficient when there is heteroskedasticity.

Because the heteroskedasticity depends on  $x_2$  and not on  $x_3$ , the estimates of  $\beta_{q2}$  vary over the quantiles  $q$ , while  $\beta_{q3}$  is invariant with respect to  $q$ . The DGP implies that the coefficient of  $x_2$  equals  $1 + 0.5 \times F_e^{-1}(0.75)$ , where  $e \sim N(0, 5^2)$ .

```

. * Predicted coefficients of x2 from quantile regression
. qui summarize e, detail
. display "Predicted coefficient of x2 for q = .25, .50, and .75" _newline
>      "      are      " 1+.5*5*invnormal(0.25) ", " 1+.5*5*invnormal(0.5)
>      ", and " 1+.5*5*invnormal(0.75)
Predicted coefficient of x2 for q = .25, .50, and .75
      are      -.68622438, 1, and 2.6862244

```

For  $q = 0.75$ , for example, the estimated coefficient of  $x_2$  is 2.573, close to the theoretical value of 2.686.

We can test the difference in CQR estimates by using the `bsqreg` command. A test of  $\beta_{0.25,2} = \beta_{0.75,2}$  can be interpreted as a robust test of heteroskedasticity independent of the functional form of the heteroskedasticity. The test is implemented as follows:

```

. * Test equality of coeff of x2 and equality of coeff of x3 for q=.25 and q=.75
. set seed 10101
. qui sqreg y x2 x3, q(.25 .75) reps(400)
. test [q25]x2 = [q75]x2
( 1)  [q25]x2 - [q75]x2 = 0
      F( 1,  9997) = 1450.56
      Prob > F =     0.0000
. test [q25]x3 = [q75]x3
( 1)  [q25]x3 - [q75]x3 = 0
      F( 1,  9997) =     0.67
      Prob > F =     0.4128

```

The test outcome leads to a strong rejection of the hypothesis that  $x_2$  does not affect both the location and scale of  $y$ . As expected, the test for  $x_3$  yields a  $p$ -value of 0.41, which does not lead to a rejection of the null hypothesis.

## 15.5 Quantile treatment effects for a binary treatment

The treatment evaluation literature considers in detail the effect on an outcome of interest of a binary treatment. The simplest and cleanest example is an experiment with random assignment of individuals to either a treated group or a control group. Basic difference-in-means analysis calculates the average treatment effect  $\bar{y}_{\text{treat}} - \bar{y}_{\text{control}}$  and tests whether this is statistically different from zero. A richer analysis estimates the difference at various quantiles, rather than the difference in means.

Consider a binary treatment  $D$  that takes values 1 if treated and 0 if not treated. The quantile treatment effect (QTE) of  $D$  at a selected quantile  $q$  is a measure of the shift in the distribution of  $y$  resulting from a unit shift in the treatment variable. That is,

$$\Delta_q = Q_q(Y|D = 1) - Q_q(Y|D = 0)$$

As an example, consider the effect on the quantiles of `ltotexp` on whether the person has supplementary health insurance. Quantiles for those with or without supplementary insurance can be obtained using the `centile` command or, equivalently, using CQR regression on a constant.

For example, separate estimation of the 25th quantile by value of `suppins` yields

```

. * QTE at q=0.25 by separate estimation for suppins==0 and suppins==1
. qui use mus203mepsmedexp, clear
. qui drop if ltotexp == .
. qreg ltotexp if suppins==0, quantile(0.25) vce(robust) nolog
.25 Quantile regression                               Number of obs =      1,207
  Raw sum of deviations   559.861 (about 7.0282016)
  Min sum of deviations   559.861                               Pseudo R2      =      0.0000

```

ltotexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
_cons	7.028202	.055348	126.98	0.000	6.919613	7.136791

```

. scalar q25_0 = _b[_cons]
. qreg ltotexp if suppins==1, quantile(0.25) vce(robust) nolog
.25 Quantile regression                               Number of obs =      1,748
  Raw sum of deviations   720.714 (about 7.4235682)
  Min sum of deviations   720.714                               Pseudo R2      =      0.0000

```

ltotexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
_cons	7.426549	.0403812	183.91	0.000	7.347348	7.505749

```

. scalar q25_1 = _b[_cons]
. display "QTE of suppins at q=25 = " q25_1 - q25_0
QTE of suppins at q=25 = .39834738

```

The QTE at  $q = 0.25$  is the difference  $7.426549 - 7.028202 = 0.398347$ , which is quite substantial.

The same QTE estimate can be obtained more simply by CQR of `ltotexp` on an intercept and `suppins`.

```

. * QTE at q=0.25 by direct CQR on intercept and suppins
. qreg ltotexp suppins, quantile(0.25) vce(robust) nolog
.25 Quantile regression                               Number of obs =      2,955
  Raw sum of deviations 1293.577 (about 7.2675252)
  Min sum of deviations 1280.575                               Pseudo R2      =      0.0101

```

ltotexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
suppins	.3983474	.0700433	5.69	0.000	.2610088	.535686
_cons	7.028202	.0569596	123.39	0.000	6.916517	7.139886

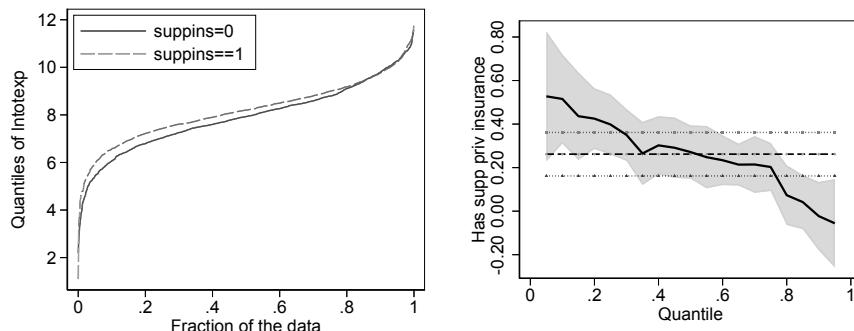
Note that the QTE is not measuring the impact of supplementary insurance for a given person. That is, we do not assume that a person who is in the 25th percentile without supplementary insurance would also be in the 25th percentile if he or she instead had supplementary insurance, a strong restriction called rank invariance or rank preservation, which is discussed further in section 25.8.4.

A useful way to present results for a range of quantiles is to use graphical displays.

We first obtain separate quantile plots by the two values of `suppins` using the community-contributed `qplot` command and overlay these on the same graph.

```
. * Quantile plot of ltotexp for those with and without supplementary insurance
. qplot ltotexp, over(suppins) clp(1 _) recast(line) scale(1.1)
>     ytitle("Quantiles of ltotexp") xtitle("Fraction of the data")
>     legend(pos(11) ring(0) col(1))
>     legend(label(1 "suppins=0") label(2 "suppins==1"))
```

The first panel of figure 15.4 gives the plots. The vertical distance between the two curves at a given value of  $q$  measures the unconditional ME of insurance. For example, at  $q = 0.25$ , we have the values 7.43 and 7.03 that were obtained earlier. Over most of the range of log(expenditure), having supplementary insurance is associated with higher expenditure; an exception occurs around the 90th percentile. The difference is greatest at lower quantiles.



**Figure 15.4.** Quantile and density plots of log expenditure for those with and without supplementary insurance

The QTE at various quantiles can be plotted using `grqreg` following CQR of `ltotexp` on an intercept and `suppins`.

```
. * Plot of the QTE of supplementary insurance at different quantiles  
. set seed 10101  
. qui bsqreg ltotexp suppins, reps(400)  
. grqreg, ci ols olsci scale(1.1) seed(10101)
```

The second panel of figure 15.4 gives the plot. The solid line gives the QTES, and the shaded area gives 95% confidence intervals. For example, at  $q = 0.25$  the solid line has value 0.40, as obtained earlier. It is clear that the QTE is much larger at the lowest quantiles and is actually slightly negative at the highest quantile. The dashed lines give the OLS estimate, which is the average treatment effect, along with 95% confidence intervals.

[Bitler, Gelbach, and Hoynes \(2006\)](#) provide an excellent illustration of this approach applied to an experiment with random assignment. Extension to nonexperimental settings such as the example of this chapter, however, is not straightforward.

First, the QTES in our example can be given only a causal interpretation if supplementary insurance was randomly assigned. Random assignment allows the distribution of `ltotexp` for those without supplementary insurance to also be used as the distribution of `ltotexp` for those with supplementary insurance in the counterfactual situation of their not having supplementary insurance.

Second, to obtain QTES with a causal interpretation, we need to control for the endogeneity of `suppins`. A simple regression approach assumes it is sufficient to add additional control variables and estimate by CQR methods already presented. An instrumental variables approach uses instruments for `suppins`. Several methods for quantile instrumental variables have been proposed, based on different assumptions.

Finally, we may ultimately be interested in the impact of `suppins` on the marginal distribution or unconditional distribution of `ltotexp`, rather than on the distribution of `ltotexp` after conditioning on variables such as `totchr`, `age`, ....

These complications are handled by quantile instrumental variables and by unconditional QR methods presented in chapter 25.

## 15.6 Additional resources

The Stata commands related to `qreg` are `bsqreg`, `iqreg`, `sqreg`, and the community-contributed command `qreg2`. See [R] **qreg** and [R] **qreg postestimation** for details of Stata's official CQR commands. The community-contributed `qplot` command is illustrated by its author in some detail in [Cox \(2005\)](#). The community-contributed `grqreg` command was created by [Azevedo \(2004\)](#).

## 15.7 Exercises

1. Consider the medical expenditures data example of section [15.3](#), except use `totexp` rather than `ltotexp` as the dependent variable. Use the same sample, so still drop if `ltotexp==..`. Estimate the parameters of the model with  $q = 0.5$  using `qreg`, and comment on the parameter estimates. Reestimate using `bsqreg` and compare results. Use `sqreg` to estimate at the quantiles 0.25, 0.50, and 0.75. Compare these estimates with each other (and their precision) and also with OLS (with robust standard errors). Use the community-contributed `grqreg` command after `bsqreg` to further compare estimates as `qreg` varies.
2. Use the medical expenditures data of section [15.3](#). Show that the median of `ltotexp` equals the exponential of the median of `totexp`. Now, add a single regressor, the indicator `female`. Then any conditional quantile function must be linear in the regressor, with  $Q_q(\ln y|\mathbf{x}) = \alpha_{q1} + \alpha_{q2}\text{female}$  and  $Q_q(y|\mathbf{x}) = \beta_{q1} + \beta_{q2}\text{female}$ . Show that if we estimate `qreg ltotexp female`, then predict, and finally exponentiate the prediction, we get the same prediction as that directly from `qreg totexp female`. Now, add another regressor, say, `totchr`. Then the conditional quantile may no longer be linear in `female` and `totchr`. Repeat the prediction exercise, and show that the invariance under the transformation property no longer holds.
3. Use the medical expenditures data example of section [15.3](#) with the dependent variable `ltotexp`. Test the hypothesis that heteroskedasticity is a function of the single variable `totchr`, which measures the number of chronic conditions. Record the test outcome. Next, test the hypothesis that the location and scale of the dependent variable expenditures vary with `totchr`. What is the connection between the two parts of this question?
4. Use the medical expenditures data of section [15.3](#), and estimate the parameters of the model for `ltotexp` using `qreg` with  $q = 0.5$ . Then, estimate the same parameters using `bsqreg` with the number of bootstrap replications set at 10, 50, 100, and 400. In each case, use the same seed of 10101. Would you say that a high number of replications produces substantially different standard errors?
5. Consider the heteroskedastic regression example of section [15.4](#). Change the specification of the variance function so that the variance function is

a function of  $x_3$  and not  $x_2$ ; that is, reverse the roles of  $x_2$  and  $x_3$ . Estimate QRS for the generated data for  $q = 0.25, 0.50$ , and  $0.75$ . Compare the results you obtain with those given in section [15.3.2](#). Next, vary the coefficient of  $x_3$  in the variance function, and study its impact on the QR estimates.

6. Consider the heteroskedasticity example of section [15.4](#). There the regression error is symmetrically distributed. Suppose we want to study whether the differences between OLS and QR results are sensitive to the shape of the error distribution. Make suitable changes to the simulation data, and implement an analysis similar to that in section [15.4](#) with asymmetric errors. For example, first draw  $u$  from the uniform distribution, and then apply the transformation  $-\lambda \log(u)$ , where  $\lambda > 0$ . (This generates draws from the exponential distribution with a mean of  $\lambda$  and a variance of  $\lambda^2$ .)
7. When the number of regressors in the QR is very large and one wants to generate graphs only for selected coefficients, it may be necessary to write one's own code to estimate and save the coefficients. This would be followed by a suitable twoway plot. The following program uses `postfile` to save the output in `bsqrcoef1.dta`, `forvalues` to loop around values of  $q = 0.1(0.1)0.9$ , and `bsqreg` to estimate bootstrap standard errors. Run the following program, and use `bsqrcoef1.dta` and the `twoway` command to generate a plot for the coefficient of `suppins` as  $q$  varies.

```

* Save coefficients, and generate graph for a range of quantiles
use mus03data.dta, clear
drop if ltotexp == .
capture program drop mus07plot
program mus07plot
postfile myfile percentile b1 upper lower using bsqrcoef1.dta, replace
forvalues tau1=0.10(0.1)0.9 {
    set seed 10101
    quietly bsqreg ltotexp suppins age female white totchr,      ///
        quantile(`tau1') reps(400)
    matrix b = e(b)
    scalar b1=b[1,1]
    matrix V = e(V)
    scalar v1=V[1,1]
    scalar df=e(df_r)
    scalar upper = b1 + invtail(df,.025)*sqrt(v1)
    scalar lower = b1 - invtail(df,.025)*sqrt(v1)
    post myfile (`tau1') (b1) (upper) (lower)
    matrix drop V b
    scalar drop b1 v1 upper lower df
}
postclose myfile
end
mus07plot
program drop mus07plot
use bsqrcoef1.dta, clear
twoway connected b1 percentile || line upper percentile ||      ///
    line lower percentile,                                         ///
    title("Slope estimates") subtitle("Coefficient of suppins")  ///
    xtitle("Quantile", size(medlarge))                         ///
    ytitle("Slope and confidence bands", size(medlarge))       ///
    legend( label(1 "Quantile slope coefficient")               ///
    label(2 "Upper 95% bs confidence band")                   ///
    label(3 "Lower 95% bs confidence band"))                  ///
graph save bsqrcoef1.gph, replace

```

8. Reconsider the analysis in section [15.3](#). Omit the variable `female` from the QR. Instead, run the regression separately for males and females for  $q = 0.25$  and  $q = 0.75$ . Compare the results and comment on differences, if any.

# **Appendix A**

## **Programming in Stata**

In this appendix, we build on the introduction to Stata programming given in chapter [1](#). We first present Stata matrix commands, introduced in section [1.5](#). The rest of the appendix focuses on aspects of writing Stata programs, using the `program` command introduced in section [5.3.1](#). We discuss programs to be included within a Stata do-file, ado-files, which are programs intended to be used by other Stata users, and some tips for program debugging that are relevant for even the simplest Stata coding.

## A.1 Stata matrix commands

Here we consider Stata matrix commands, initiated with the `matrix` prefix. These provide a limited set of matrix commands sufficient for many uses, especially postestimation manipulation of results, as introduced in section [1.6](#), and are comparable with matrix commands provided in other econometrics packages.

The separate appendix [B](#) presents Mata matrix commands, introduced in Stata 9. Mata is a full-blown matrix programming language, comparable with R, MATLAB, or Gauss.

### A.1.1 Stata matrix overview

Key considerations are inputting matrices, either directly or by converting data variables into matrices, and performing operations on matrices or on subcomponents of the matrix such as individual elements.

The basics are given in [U] **14 Matrix expressions** and in [P] **matrix**. Useful online help commands include `help matrix`, `help matrix operators`, and `help matrix functions`.

### A.1.2 Stata matrix input and output

There are several ways to input matrices in Stata.

#### Matrix input by hand

Matrix entries can be entered by using the `matrix define` command. For example, consider a  $2 \times 3$  matrix with the first row entries 1, 2, and 3 and the second row entries 4, 5, and 6. Column entries are separated by commas, and rows are separated by a backslash. We have

```

. * Define a matrix explicitly and list the matrix
. matrix define A = (1,2,3 \ 4,5,6)
. matrix list A
A[2,3]
    c1   c2   c3
r1    1    2    3
r2    4    5    6

```

The word `define` can be omitted from the above command.

The default names for the matrix rows are `r1`, `r2`, ..., and the column defaults are `c1`, `c2`, .... These names can be changed by using the `matrix rownames` and `matrix colnames` commands. For example, to give the names `one` and `two` to the two rows of matrix `A`, type the command

```

. * Matrix row and column names
. matrix rownames A = one two
. matrix list A
A[2,3]
    c1   c2   c3
one   1    2    3
two   4    5    6

```

An alternative matrix naming command is `matname`.

### **Matrix input from Stata estimation results**

Matrices can be constructed from matrices created by the Stata estimation command results stored in `e()` or `r()`. For example, after ordinary least-squares (OLS) regression, the variance–covariance matrix is stored in `e(V)`. To give it a more obvious name or to save it for later analysis, we define a matrix equal to `e(V)`.

As a data example, we use the same dataset as in chapter 3. We use the first 100 observations and regress medical expenditures (`ltotexp`) on an intercept and chronic problems (`totchr`). We have

```

. * Read in data, summarize, and run regression
. use mus203mepsmedexp
(A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e)
. keep if _n <= 100
(2,964 observations deleted)
. drop if ltotexp == . | totchr == .
(0 observations deleted)
. summarize ltotexp totchr

```

Variable	Obs	Mean	Std. dev.	Min	Max
ltotexp	100	4.533688	.8226942	1.098612	5.332719
totchr	100	.48	.717459	0	3

```

. regress ltotexp totchr, noheader

```

	Coefficient	Std. err.	t	P> t	[95% conf. interval]
ltotexp	.1353098	.1150227	1.18	0.242	-.0929489 .3635685
_cons	4.468739	.0989462	45.16	0.000	4.272384 4.665095

A command to drop observations with missing values from the dataset in memory is included because not all matrix operators considered below handle missing values.

We then obtain the variance matrix stored in `e(V)` and list its contents.

```

. * Create a matrix from estimation results
. matrix vbols = e(V)
. matrix list vbols
symmetric vbols[2,2]
      totchr      _cons
totchr  .01323021
_cons   -.0063505  .00979036

```

Stata has incorporated the regressor names into the estimate of the variance-covariance matrix of the estimator so that `vbols` has rows and columns named `totchr` and `_cons`.

### A.1.3 Stata matrix subscripts and combining matrices

Matrix subscripts are represented in square brackets. The entry  $(i, j)$  in a matrix is denoted `[i, j]`. For example, to set the  $(1, 1)$  entry in matrix A to equal the  $(1, 2)$  entry, type the command

```

. * Change value of an entry in matrix
. matrix A[1,1] = A[1,2]
. matrix list A
A[2,3]
    c1  c2  c3
one   2   2   3
two   4   5   6

```

If the row or column has a name, one can alternatively use this name. For example, because row 1 of `A` is named `one`, we could have typed `matrix A[1,1] = A["one",2]`.

For a column vector, the  $i$ th entry is denoted by `[i,1]` rather than simply `[i]`. Similarly, for a row vector, the  $j$ th entry is denoted by `[1,j]` rather than simply `[j]`.

Matrix subscripts can be given as a range, permitting a submatrix to be extracted from a matrix. For example, to extract all the rows and columns 2–3 from matrix `A`, type

```

. * Select part of matrix
. matrix B = A[1...,2..3]
. matrix list B
B[2,2]
    c2  c3
one   2   3
two   5   6

```

Here `k...` selects the  $k$ th entry on, and `k..l` selects the  $k$ th– $l$ th entry.

To add or append rows to a matrix, use the vertical concatenation operator `\`. For example, `A \ B` adds rows of `B` after the rows of `A`. Similarly, to add columns to a matrix, use the horizontal concatenation operator `,`. For example,

```

. * Add columns to an existing matrix
. matrix C = B, B
. matrix list C
C[2,4]
    c2  c3  c2  c3
one   2   3   2   3
two   5   6   5   6

```

## A.1.4 Matrix operators

All the standard matrix operators can be applied, provided that the matrices are conformable. The operators are `+` to add, `-` to subtract, `*` to multiply, and `#` for the Kronecker product. In addition, the multiplication command can be used for multiplication by a scalar, for example, `2*A` or `A*2`, and scalar division is possible, for example, `A/2`. A single apostrophe, `',`, gives the matrix transpose. To compute  $A' A$ , we use `A' *A`. For example,

```
. * Matrix operators
. matrix D = C + 3*C
. matrix list D
D[2,4]
      c2   c3   c2   c3
one    8   12    8   12
two   20   24   20   24
```

## A.1.5 Matrix functions

Standard matrix functions are defined by using parentheses, `()`. Some commands lead to a scalar result, for example,

```
. * Matrix functions
. matrix r = rowsof(D)
. matrix list r
symmetric r[1,1]
      c1
r1    2
```

In this example, it is more convenient to store the result as a scalar, rather than in a  $1 \times 1$  matrix. For example,

```
. * Can use scalar if 1 x 1 matrix
. scalar ralt = rowsof(D)
. display ralt
2
```

Functions that produce scalars include `colsof(A)`, `det(A)`, `rowsof(A)`, and `trace(A)`.

Other commands produce matrices. For example, matrix `B`, created earlier, is a nonsymmetric square matrix, with the inverse

```
. * Inverse of nonsymmetric square matrix
. matrix Binv = inv(B)
. matrix list Binv
Binv[2,2]
      one          two
c2       -2           1
c3    1.6666667  -.6666667
```

Here are some functions that produce matrices: `cholesky(A)`, `corr(A)`, `diag(A)`, `hadamard(A,B)`, `I(n)`, `inv(A)`, `invsym(A)`, `vec(A)`, and `vecdiag(A)`.

### A.1.6 Matrix accumulation commands

Most estimators, such as the OLS estimator  $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ , require the computation of matrix cross products. We strongly recommend that you do not put your data into Stata matrices. Stata has accumulation commands that compute cross products from variables and return the results in Stata matrices. If you really want to put your data into a matrix, refer to appendix [B](#) on Mata.

Stata's matrix accumulation commands compute the matrix cross products  $\mathbf{X}'\mathbf{X}$  and  $\mathbf{X}'\mathbf{y}$  without requiring the intermediate step of forming the much larger matrices  $\mathbf{X}$  and  $\mathbf{y}$ .

As an example, the `matrix accum A = v1 v2` command creates a  $3 \times 3$  matrix  $\mathbf{A} = \mathbf{Z}'\mathbf{Z}$ , where  $\mathbf{Z}$  is an  $N \times 3$  matrix with columns of the variables `v1` and `v2` and a column of 1s that `accum` automatically appends unless the `noconstant` option is used. The companion `matrix vecaccum A = w v1 v2` command creates a  $1 \times 3$  row vector  $\mathbf{A} = \mathbf{w}'\mathbf{Z}$ , where  $\mathbf{w}$  is an  $N \times 1$  column vector with a column of the variable `w` and  $\mathbf{Z}$  is an  $N \times 3$  matrix with columns of the variables `v1` and `v2` and a column of 1s that again `accum` automatically adds at the end unless the `noconstant` option is used. Related commands are `matrix glsaccum`, which forms weighted cross products of the form  $\mathbf{X}'\mathbf{B}\mathbf{X}$ , and `matrix opaccum`.

The following code produces the same point estimates as `regress`

```
ltotexp totchr:  
  
. * OLS estimator using matrix accumulation operators  
. matrix accum XTX = totchr // Form X'X including constant  
(obs=100)  
. matrix vecaccum yTX = ltotexp totchr // Form y'X including constant  
. matrix cols = invsym(XTX)*(yTX)'  
. matrix list cols  
cols[2,1]  
    ltotexp  
totchr  .13530976  
_cons   4.4687394
```

### A.1.7 OLS using Stata matrix commands

The following example runs an OLS regression of `ltotexp` on an intercept and `totchr`, and it also reports the default OLS standard errors and associated  $t$  statistics. We use matrix accumulation commands so that large problems can be handled. The challenge in using these commands is to obtain  $s^2 = \sum_{i=1}^N (y_i - \hat{y}_i)^2$  without having to form an  $N \times 1$  vector of predicted values. One way to do this is to use the result that for OLS  $\sum_{i=1}^N (y_i - \hat{y}_i)^2 = \mathbf{y}'\mathbf{y} - \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}}$ .

We have

```

. * Illustrate Stata matrix commands: OLS with output
. matrix accum XTX = totchr          // Form X'X including constant
(obs=100)
. matrix vecaccum yTX = ltotexp totchr // Form y'X including constant
. matrix b = invsym(XTX)*(yTX) '
. matrix accum yTy = ltotexp, noconstant
(obs=100)
. scalar k = rowsof(XTX)
. scalar n = _N
. matrix s2 = (yTy - b'*XTX'*b)/(n-k)
. matrix V = s2*invsym(XTX)
. matrix list b
b[2,1]
      ltotexp
totchr  .13530976
_cons   4.4687394
. matrix list V
symmetric V[2,2]
      totchr      _cons
totchr  .01323021
_cons   -.0063505  .00979036

```

This yields the same estimates of the coefficients and variance–covariance matrix of the estimator as listed in appendix [A.1.2](#).

We now want to obtain output formatted in the usual way with columns of regressor names, coefficient estimates, standard errors, and  $t$  statistics. This is not straightforward using Stata matrix commands. We wish to form the column vector  $\tau$  with the  $j$ th entry  $t_j = b_j/s_j = b_j/\sqrt{V_{jj}}$ . But Stata provides no facility for element-by-element division and also no easy way to take the element-by-element square root of a matrix. One fix is to first form a column vector, `seinv`, with the  $j$ th entry  $1/s_j$  by creating a diagonal matrix with the entries  $s_j^2$ , taking the inverse of this matrix, taking the square root of this matrix, and forming a column vector with the resulting diagonal entries. Then, form the column vector  $\tau$  by using the Hadamard product of `b` and `seinv`, where for matrices `A` and `B` of the same dimension, `C=hadamard(A,B)` gives the matrix `C` with the  $ij$ th entry  $C_{ij} = A_{ij} \times B_{ij}$ . We obtain

```

. * Stata matrix commands to compute standard errors and t statistics
> given b and V
. matrix se = (vecdiag(cholesky(diag(vecdiag(V)))))'
. matrix seinv = (vecdiag(cholesky(invsym(diag(vecdiag(V))))))'
. matrix t = hadamard(b,seinv)
. matrix results = b, se, t
. matrix colnames results = coeff serror tratio
. matrix list results, format(%7.0g)
results[2,3]
      coeff    serror    tratio
totchr   .13531    .11502    1.1764
_cons    4.4687    .09895   45.163

```

It is much easier to instead use Stata's `ereturn` commands to produce this output, based on a row vector of coefficient estimates and an estimated variance matrix. The preceding code led to a column vector of coefficients, so we first need to transpose. We obtain

```

. * Easier is to use ereturn post and display given b and V
. matrix brow = b'
. ereturn post brow V
. ereturn display

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
totchr	.1353098	.1150227	1.18	0.239	-.0901305 .36075
_cons	4.468739	.0989462	45.16	0.000	4.274808 4.66267

Similar code for OLS that instead uses Mata functions is provided in section [3.9](#).

## A.2 Programs

Do-files, ado-files, and program files are collections of Stata commands that are useful whenever the same analysis is to be repeated exactly or with relatively minor variation. For many analyses, a do-file that enacts Stata commands (which are themselves often ado-files written in Stata or Mata) is sufficient.

More advanced analysis, however, may require actual programming in Stata. These programs can be defined and executed as a component of a do-file, or they can be converted to an ado-file to enable their being called by other programs. Useful references are [U] **18 Programming Stata** and [P] **program**.

### A.2.1 Simple programs (no arguments or access to results)

A program is defined by using the `program define` command followed by the program name. Subsequent lines give the program, which concludes with the line `end`.

The simplest programs do not have any inputs, and the program output is simply displayed. The following program displays the current time and date.

```
. * Program with no arguments
. program define time
1. display c(current_time) c(current_date)
2. end
```

The word `define` is optional in the above input—it is sufficient to simply type `program time`.

The program is executed by typing the name of the program. We have

```
. * Run the program
. time
12:29:29 1 Nov 2021
```

Unlike the execution of a do-file, only the program results are listed, here the current date and time; the program commands that were executed are not

listed.

### A.2.2 Modifying a program

Stata does not allow one to redefine an existing program. So it is necessary to first remove any previous program with the same name, should such a program already exist.

The `program drop time` command will drop the `time` program. If this program does not already exist, however, Stata will stop executing and generate an error message. The `capture` prefix ensures that Stata will continue to run, even if the `time` program does not already exist.

Thus, the preferred way to define and then execute the `time` program is

```
. * Drop program if it already exists, write program, and run
. capture program drop time
. program time
  1. display c(current_time) c(current_date)
  2. end
. time
12:29:29 1 Nov 2020
```

The `clear` command does not drop programs, though `clear all` will. To specifically drop all programs, use the `clear programs` command or the `program drop _all` command.

### A.2.3 Programs with positional arguments

More complicated programs have inputs called arguments. For example, the Stata `regress` command has as arguments the dependent variable and any regressor variables. Then execution of the command requires that one give both the command name and the command arguments, for example, `regress y x1 x2`. These arguments need to be passed into the program and then referred to appropriately within the program.

Program arguments can be passed as positional arguments. The first argument is referred to by the local macro '`1`' within the program, the second by local macro '`2`', and so on. For example, for `regress`, the

dependent variable, say, `y`, may be referred to internally as ‘`1`’. The quotation marks differ from how they appear on this printed page. On most keyboards, the left quote is located in the upper left, and the right quote is located in the middle right. When viewed using a text editor, the single quote appears correctly. But often when viewed in LaTeX documents, they misleadingly appear as ‘`1`’ rather than the correct ‘`1`’.

We present a program to report the median of the difference between two variables, where the two variables need to be passed to the program. Using positional arguments, we have

```
. * Program with two positional arguments
. program meddiff
1.      tempvar diff
2.      generate `diff' = `1' - `2'
3.      _pctile `diff', p(50)
4.      display "Median difference = " r(r1)
5. end
```

The program uses a temporary variable, `diff`, explained in the next section, to store the difference between the two variables. Several commands calculate the median. Here we use the `_pctile` command with the `p(50)` option. This command stores the resulting median in `r(r1)`, which we then output by using the `display` command.

We now run the `meddiff` program, using the same dataset and variables, `ltotchr` and `totchr`, as used in appendix [A.1](#). We have

```
. * Run the program with two arguments
. meddiff ltotexp totchr
Median difference = 4.2230513
```

#### A.2.4 Temporary variables

The `meddiff` program requires the computation of the intermediate variable we have named `diff`. To ensure that this name does not conflict with the names of variables elsewhere and that the variable is dropped as soon as the program ends, we use the `tempvar` command to define a temporary variable that is local only to the program and is dropped after the program has executed. This temporary variable is declared by using `tempvar` and is then referred to in the same left and right quotation marks as are used for local

macros. Similarly, the `tempname` command can be used to declare temporary scalars and matrices, and the  `tempfile` command can be used to declare temporary files.

### A.2.5 Programs with named positional arguments

It is much easier to read the program if it gives names to the positional arguments '1', '2', .... To use named positional arguments, we first define the arguments within the program in the order that they will appear in the command. For example,

```
. * Program with two named positional arguments
. capture program drop meddiff
. program meddiff
1.     args y x
2.     tempvar diff
3.     generate `diff' = `y' - `x'
4.     _pctile `diff', p(50)
5.     display "Median difference = " r(r1)
6. end
. meddiff ltotexp totchr
Median difference = 4.2230513
```

As for temporary variables, the arguments are declared without quotes, but Stata stores arguments as local macros, so we need to use quotes to refer to the arguments.

### A.2.6 Storing and retrieving program results

The preceding examples simply displayed results. Often, we want to store program results for further data analysis. This can be done by storing the results in `r()` and `e()`, introduced in section [1.6](#), and `s()`. To do this, we need to define the program to be of the relevant class and to return the results to the named entries in `r()`, `e()`, or `s()`.

For our example, we declare the program to be `rclass`, with just one result that will be stored in `r(medylx)`. We have

```

. * Program with results stored in r()
. capture program drop meddiff
. program meddiff, rclass
  1. args y x
  2. tempvar diff
  3. generate `diff' = `y' - `x'
  4. _pctile `diff', p(50)
  5. return scalar medylx = r(r1)
6. end

```

Executing the program produces no output; the results of executing the program are instead stored in `r()`. To list the program results in `r()`, we use the `return list` command, and to display the scalars in `r()`, we use the `display` command.

```

. * Running the program does not immediately display the result
. meddiff ltotexp totchr
. return list
scalars:
      r(medylx) =  4.223051309585571
. display r(medylx)
4.2230513

```

An example of an `e`class program, returning `b` and `v` for subsequent analysis, is given in section [12.4.4](#).

### A.2.7 Programs with arguments using standard Stata syntax

Program arguments can be quite lengthy and can include optional arguments, but many commands use arguments in a standard format. In particular, if commands use the standard Stata syntax, then tools exist to parse the command, breaking down the long command into its various arguments.

The full standard Stata syntax is

$$\text{command} \left[ \text{varlist} \mid \text{namelist} \mid \text{anything} \right] \left[ \text{if} \right] \left[ \text{in} \right] \left[ \text{using filename} \right] \left[ =\text{exp} \right] \left[ \text{weight} \right] \left[ , \text{options} \right]$$

Brackets denote optional items. For some commands, some of the items in brackets will be required to run that specific command. In the syntax for that specific command, these required items will not be surrounded by brackets.

As an example, consider the command

```
. regress ltotexp totchr if !missing(ltotexp) in 1/100, vce(robust)
```

To enact this command, Stata interprets `regress` as *command*, `ltotexp` `totchr` as *varlist*, `if !missing(ltotexp)` as *if*, `in 1/100` as *in*, and `vce(robust)` as an *option*. For the `regress` command, Stata needs to further break down *varlist*, with the first variable being the dependent variable and any remaining variables being regressors.

We now show how this is done. We write a program, `myols`, that duplicates `regress`. Specifically, we want to be able to break the command

```
. myols ltotexp totchr if !missing(ltotexp) in 1/100, vce(robust)
```

into its arguments and then execute `regress` with these arguments.

To do so, we use the `syntax` and `gettoken` commands as illustrated in the following program.

```
. * Program that uses Stata commands syntax and gettoken to parse arguments
. program myols
1.   syntax varlist [if] [in] [,vce(string)]
2.   gettoken y xvars : varlist
3.   display "varlist contains: " ``varlist''
4.   display "and if contains: " ``if''
5.   display "and in contains: " ``in''
6.   display "and vce contains: " ``vce''
7.   display "and y contains: " ``y''
8.   display "& xvars contains: " ``xvars''
9.   regress `y' `xvars' `if' `in', `vce' noheader
10.  end
```

The `syntax` command lists required arguments—which here are a list of variable names (`varlist`)—and optional arguments—which here are an `if` qualifier, an `in range` qualifier, and the `vce()` option with a *string* argument for the specific option to be used (`[, vce(string)]`). The `syntax` command will put in the local '`varlist`' macro any list of variable names that appears after `myols` and before the `if` or `in` qualifiers; in the local '`if`' macro any `if` qualifier; in the local '`in`' macro any `in range` qualifier; and in the local '`vce`' macro any `vce_option`. The names in '`varlist`' are space-separated tokens. The specific form of the `gettoken` command used here puts the first

token in ‘varlist’ into the local ‘y’ macro and the remaining tokens into the local ‘xvars’ macro.

The unnecessary `display` commands are included to demonstrate that the parsing occurs as desired. Note the use of compound quotes. For example, to display the name in the local ‘y’ macro, we use the `display " `y' "` command. If instead we use `display `y'`, then we would see the value of the variable in the local ‘y’ macro.

We then execute the `myols` program for an example.

```
. * Execute program myols for an example
. myols ltotexp totchr if !missing(ltotexp) in 1/100, vce(robust)
varlist contains: ltotexp totchr
and if contains: if !missing(ltotexp)
and in contains: in 1/100
and vce contains: robust
and y contains: ltotexp
& xvars contains: totchr
```

ltotexp	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
totchr	.1353098	.1089083	1.24	0.217	-.0808151	.3514347
_cons	4.468739	.1089425	41.02	0.000	4.252547	4.684932

The arguments of the `myols` command have been parsed successfully, leading to the expected output from `regress`.

## A.2.8 Ado-files

Some Stata commands, such as `summarize`, are built-in commands. But many Stata commands are defined by an ado-file, which is a collection of Stata commands. For example, the file `logit.ado` defines the `logit` command for logit regression. Furthermore, Stata users can also define their own Stata commands by using ado-files. We use many such community-contributed commands throughout this book.

An ado-file is a program file similar to those already presented. But because they are intended for wider use, they are generally more tightly written. Temporary variables, scalars, and matrices are used to avoid

potential name conflicts with the program calling the ado-file. Variables may be generated in double precision. Care is given to the output from the program, such as by using the `quietly` prefix to suppress the unnecessary printing of intermediate results. Comments are provided, such as the current version number and date. And there should be various checks to ensure that the command is being correctly used (for example, if an input to the program should be positive, then send an error message if this is not the case).

A good example of the development of an ado-file is given in [U] **18.11 Ado-files**. For an estimation command, see [Gould, Pitblado, and Poi \(2010\)](#), chap. 10).

Here we provide a brief example, converting the `meddiff` program from earlier into an ado-file. Specifically, the `meddiff.ado` file comprises

```
. capture program drop meddiff
. *! version 2.1.0 15aug2021
. program meddiff, rclass
  1.      version 17
  2.      args y x
  3.      tempvar diff
  4.      qui {
  5.          generate double `diff' = `y' - `x'
  6.          _pctile `diff', p(50)
  7.          return scalar medylx = r(r1)
  8.      }
  9.      display "Median of first variable - second variable = " r(r1)
10. end
```

The program begins with the version and date. The program is written for Stata 17. The `quietly` prefix suppresses output. For example, if '`y`' or '`x`' has any missing values, then the `generate` statement will lead to a statement that missing values were generated. This statement will be suppressed here. The '`diff`' variable is in double precision for increased accuracy.

To execute the commands in `meddiff.ado`, we simply type `meddiff` with the appropriate arguments. For example,

```
. * Execute program meddiff for an example
. meddiff ltotexp totchr
Median of first variable - second variable = 4.2230513
```

The `meddiff.ado` file needs to be in a directory that Stata automatically accesses. For a Microsoft Windows computer, these directories include `C:\ado` and `C:\Program Files\Stata17` and the current directory. See [U] **17 Ado-files** for further details.

## A.3 Program debugging

This section provides advice relevant to even the most basic uses of Stata.

There are two challenges: to get the program to execute without stopping because of an error and to ensure that the program is doing what is intended once it is executing.

We focus here on the first challenge. The simplest way to debug a program is to work with a simplified example and print out intermediate results. Stata also provides error messages and a trace facility to track every step of the execution of a program.

The second challenge is easily ignored, but it should not be skipped. Come up with an example where there is a known result or a way to verify the result. For example, to test an estimation procedure, generate many observations from a known data-generating process, and see whether the estimation procedure yields the known data-generating process parameters; see section [5.6](#). Printing intermediate results is again very helpful. In particular, always use the `summarize` command to verify that you are working with the intended dataset.

### A.3.1 Some simple tips

A simple way to debug Stata code is to display the intermediate output. For example, in the following listing, we can see whether the correct dimension matrices are obtained. If the program failed, we could look at the intermediate results before the failure to see where the failure occurs.

```

. * Display intermediate output to aid debugging
. matrix accum XTX = totchr          // Recall constant is added
(obs=100)
. matrix list XTX                  // Should be 2 x 2
symmetric XTX[2,2]
    totchr      _cons
totchr      74
_cons       48      100
. matrix vecaccum yTX = ltotexp totchr
. matrix list yTX                  // Should be 1 x 2
yTX[1,2]
    totchr      _cons
ltotexp  224.51242  453.36881
. matrix bOLS = invsym(XTX)*(yTX) '
. matrix list bOLS                // Should be 2 x 1
bOLS[2,1]
    ltotexp
totchr   .13530976
_cons    4.4687394

```

Even when there seems to be no problem, if the program is still being debugged, it can be useful to comment out an extraneous command, such as `matrix list`, rather than to delete the command, in case there is reason to use it again later.

Debugging can be quicker and simpler if one works with a simplified program. For example, rather than work with the full dataset and many regressors, one might initially work with a small subset of the data and a single regressor. This may also reduce the chance that problems are arising merely because of data problems, such as multicollinearity.

To further save time, one may find it worthwhile to use `/*` and `*/` to comment out those parts of the program that are not needed during the debugging exercise. This is especially the case for computationally intensive tasks that are not necessary, such as graphs to be used in the final analysis but not needed during the program development stage.

### A.3.2 Error messages and return code

Stata produces error messages. The message given can be brief, but a fuller explanation can be obtained from the manual or directly from Stata.

For example, if we regress `y` on `x` but one or both of these variables do not exist, we get

```
. regress y x  
variable y not found  
r(111);
```

For a more detailed explanation of the return code 111, type the command

```
. search rc 111  
(output omitted)
```

If a Stata program is being debugged, then program failure can lead to an error message that is not at all helpful. More useful error messages can be given if the code is not embedded in a program. Thus, rather than work with a program in the program environment, it can be helpful at first to work with the commands in a Stata do-file but not within a program. For example, a nonprogram version of the `meddiff` program is

```
. * Debug an initial nonprogram version of a program  
. tempvar y x diff  
. generate `y' = ltotexp  
. generate `x' = totchr  
. generate double `diff' = `y' - `x'  
. _pctile `diff', p(50)  
. scalar medylx = r(r1)  
. display "Median of first variable - second variable = " medylx  
Median of first variable - second variable = 4.2230513
```

### A.3.3 Trace

The `trace` command traces the execution of a program. To initiate a trace, type the command

```
. set trace on  
(output omitted)
```

To stop the trace, type the command

```
. set trace off
```

The `trace` facility can generate a large amount of output. Thus, it can be more useful to manually insert commands that give intermediate results. The default is `set trace off`.

## A.4 Additional resources

The [P] *Stata Programming Reference Manual* provides details. See also [Baum \(2016\)](#).

## **Appendix B**

### **Mata**

Mata is a powerful matrix programming language comparable with Gauss, MATLAB, or R. Compared with the Stata `matrix` commands, it is computationally faster; supports larger matrices (Mata has no restriction on matrix size, so the only restriction is computer specific); has a wider range of matrix commands; and has commands that are closer in syntax to the matrix notation used in mathematics.

Mata is a component of Stata that can be used on its own. Additionally, one can blend Stata and Mata functions.

## B.1 How to run Mata

Mata commands are usually run in Mata, which is initiated by first giving the `mata` command in Stata. Single Mata commands can be given in Stata, and single Stata commands can be given in Mata.

### B.1.1 Mata commands in Mata

Mata can be initiated by the Stata `mata` command. In Mata, the command prompt is a semicolon (`:`) rather than a period. Mata commands are separated by line breaks or by semicolons. To exit Mata and return to Stata, use the Mata `end` command.

The following sample Mata session creates a  $2 \times 2$  identity matrix, `I`, and then displays the elements of matrix `I`.

```
. * Sample Mata session
. mata
: I = I(2)                                              mata (type end to exit)
: I
[symmetric]
      1   2
  1 |  1
  2 |  0   1
: end
```

For symmetric matrices, such as the identity matrix, only the lower triangle is listed. Here the unlisted (1, 2) element equals the listed (2, 1) element, which is 0.

### B.1.2 Mata commands in Stata

A single Mata command can be issued in Stata by adding the `mata` prefix before the Mata command.

For example, to create a  $2 \times 2$  identity matrix,  $\mathbb{I}$ , and to display the elements of  $\mathbb{I}$ , type the commands

```
. * Mata commands issued from Stata
. mata: I = I(2)
. mata: I
[symmetric]
    1   2
  1  1
  2  0   1
```

### B.1.3 Stata commands in Mata

Mata commands are distinct from Stata commands. One can enact a Stata command within a Mata program, however, by using the `stata()` function within Mata.

For example, suppose we are in Mata and want to find the mean of the `ltotexp` variable, which is in the Stata dataset currently in memory. In Stata, we would type the `summarize ltotexp` command. In Mata, we use the `stata()` function with the desired Stata command in double quotes as the argument.

```
. // Stata commands issued from Mata
. mata
: stata("summarize ltotexp")                               mata (type end to exit)
  Variable |       Obs        Mean      Std. dev.       Min       Max
  ltotexp  |     100    4.533688    .8226942    1.098612    5.332719
: end
```

### B.1.4 Interactive versus batch use

There are differences between what is possible in Mata interactive use and what is possible in a Mata program. For example, comments cannot be included in Mata in interactive use.

### B.1.5 Mata help

We provide some basic Mata code in this appendix. The two-volume set of Mata manuals is very complete but does not provide as many data-oriented examples as appear in the other Stata manuals.

The `help` command for Mata works at either Stata's dot prompt or Mata's colon prompt.

If you know the name of the matrix command, operator, or function, then type the `help mata name` command. For example, if you know that the `det()` function takes the determinant of a matrix, then type the command

```
: help mata det  
(output omitted)
```

In this example, the command was typed in Mata, but exactly the same `help` command can be typed in Stata.

If you do not know the specific name, then it is harder. For example, suppose we want to find help on the category `matrix`. Then no specific help entry is obtained after `help mata matrix`. However,

```
: help m4 matrix  
(output omitted)
```

does work because `M-4` is the relevant section of the manuals for Mata. More generally, the command is `help m# name`, but this requires knowing the relevant section of the manuals.

Often, one must start with the `help mata` command and then selectively choose from the subsequent entries.

## B.2 Mata matrix commands

We present the various basics of creating matrices and matrix operators and functions. Explanatory comments begin with // because Mata does not recognize comments beginning with \*.

### B.2.1 Mata matrix input

#### Matrix input by hand

Matrices can be input by hand. For example, consider a  $2 \times 3$  matrix A with the first row entries 1, 2, and 3 and the second row entries 4, 5, and 6. This can be defined as follows:

```
: // Create a matrix  
: A = (1,2,3 \ 4,5,6)
```

As with the `matrix define` command in Stata, a comma is used to separate column entries, and a backslash is used to separate rows.

To see the matrix, simply type the matrix name:

```
: // List a matrix  
: A  
    1   2   3  
1  1   2   3  
2  4   5   6
```

#### Identity matrices, unit vectors, and matrices of constants

An  $n \times n$  identity matrix is created with `I(n)`. For example,

```
: // Create a 2x2 identity matrix  
: I = I(2)
```

A  $1 \times n$  row vector with 0s in all entries aside from the  $i$ th is created with `e(i, n)`. For example,

```

: // Create a 1x5 unit row vector with 1 in second entry and 0s elsewhere
: e = e(2,5)
: e
      1   2   3   4   5
1 0 1 0 0 0

```

An  $r \times c$  matrix of constants equal to the value  $v$  is created with  $\text{J}(r, c, v)$ . For example,

```

: // Create a 2x5 matrix with entry 3
: J = J(2,5,3)
: J
      1   2   3   4   5
1 3 3 3 3 3
2 3 3 3 3 3

```

Range operators create vectors with entries that increment by one for each entry by using  $a \dots b$  for a row vector and  $a :: b$  for a column vector. For example,

```

: // Create a row vector with entries 8 to 15
: a = 8..15
: a
      1   2   3   4   5   6   7   8
1 8 9 10 11 12 13 14 15

```

creates a row vector with the entries 8, 9, ..., 15.

For creation of other standard matrices, type `help m4 standard`.

### Matrix input from Stata data

Matrices can be associated with variables in the current Stata dataset in memory by using the Mata `st_view()` function.

For example, suppose the current Stata dataset includes the variables `ltotexp`, `totchr`, and `cons`. Then,

```
: // Create Mata matrices from variables stored in Stata
: st_view(y=., ., "ltotexp")
: st_view(X=., ., ("totchr", "cons"))
```

associates the column vector, `y`, with the observations on the variable `ltotexp` and a matrix, `x`, with the observations on the variables `totchr` and `cons`.

A brief summary of the syntax follows for the second `st_view()` function above. The first entry is `x=.` because this eliminates the need to previously define the vector `x`. If instead we had first entered simply `x`, we would have received the error message `<istmt>: 3499 X not found`. The second entry is a period, meaning that all the observations will be selected. The argument could instead be a list of observations. The third entry is a row vector selecting the particular variables, with variable names given in quotes and commas separating the column entries in the row vector. If `totchr` and `cons` were the 31st and 45th entries in the dataset, we could equally well type `st_view(X=., ., (31, 45))`.

The `st_view()` function creates a view of the Stata dataset that does not require that the actual data be physically loaded into Mata, saving time and memory. For example, to subsequently form the OLS estimator  $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$  in Mata, one need load only the  $K \times K$  matrix  $(\mathbf{X}'\mathbf{X})^{-1}$  and the  $K \times 1$  matrix  $\mathbf{X}'\mathbf{y}$ , not the much larger  $N \times K$  matrix  $\mathbf{X}$ .

The related `st_data()` function does actually load matrices, but this is usually not necessary. As an example,

```
. // Create a Mata matrix from variables stored in Stata
. Xloaded = st_data(., ("totchr", "cons"))
```

creates a matrix, `Xloaded`, with the  $i$ th row the  $i$ th observation on the `totchr` and `cons` variables.

### Matrix input from Stata matrix

Mata matrices can be created from matrices created by Stata commands, using the Mata `st_matrix()` function. For example,

```

: // Read Stata matrix (created in first line below) into Mata
: stata("matrix define B = I(2)")
: C = st_matrix("B")

: C
[symmetric]
      1   2
  1 | 1
  2 | 0   1

```

The `st_matrix()` function can also be used to transfer a Mata matrix to Stata; see appendix [B.2.6](#).

### Stata interface functions

Stata interface functions begin with `st_` and link matrices and data in Mata with those in Stata. Examples already given are `st_view()`, `st_data()`, and `st_matrix()`. The `st_addvar()` and `st_store()` functions are presented in appendix [B.2.6](#). A summary is given in [M-4] **Stata**, and individual `st_` functions are given in [M-5] **Intro**.

## B.2.2 Mata matrix operators

The arithmetic operators for conformable matrices are `+` to add, `-` to subtract, `*` to multiply, and `#` for the Kronecker product. The multiplication command can also be used for multiplication by a scalar, for example, `2*A` or `A*2`, and scalar division is possible, for example, `A/2`. A scalar can be raised to a scalar power, for example, `a^b`. The matrix `-A` is the negative of `A`.

A single apostrophe, `'`, gives the matrix transpose (or conjugate transpose if the matrix is complex). To compute `A' A`, we can use `A' A` or `A' * A`.

The Kronecker product of two matrices is given by `A#B`. If `A` is  $m \times n$  and `B` is  $r \times s$ , then `A#B` is  $mr \times ns$ .

### Element-by-element operators

Key arithmetic operators are the colon operators for element-by-element operations. A leading example is element-by-element multiplication of two matrices of the same dimension (the Hadamard product). Then  $C = A : * B$  has an  $ij$ th element equal to the  $ij$ th element of  $A$  times the  $ij$ th element of  $B$ .

Element-by-element multiplication of a column vector and a matrix is possible if they have the same number of rows. Similarly, element-by-element multiplication of a row vector and a matrix is possible if they have the same number of columns. For the column vector case, type

```
: // Element-by-element multiplication of matrix by column vector
: b = 2::3
: J = J(2,5,3)
: b:*J
    1   2   3   4   5
  1 | 6   6   6   6   6
  2 | 9   9   9   9   9
```

The column vector  $b$  has the entries 2 and 3, and the  $2 \times 5$  matrix  $J$  has all entries equal to 3. The first row of matrix  $J$  is multiplied by 2 (the first entry in column vector  $b$ ), and the second row of  $J$  is multiplied by 3 (the second entry in  $b$ ).

Let  $w$  be an  $N \times 1$  column vector and  $x$  be an  $N \times K$  matrix with  $i$ th row  $x'_i$ . Then  $w : * x$  is the  $N \times K$  matrix with the  $i$ th row  $w_i x'_i$ , and  $(w : * x)' x$  is the  $K \times K$  matrix equal to  $\sum_{i=1}^N w_i x_i x'_i$ .

Other colon operators are available for division ( $: /$ ), subtraction ( $: -$ ), power ( $: ^$ ), equality ( $: ==$ ), inequality ( $: !=$ ), specific inequalities (such as  $: >=$ ), “and” ( $: &$ ), and “or” ( $: |$ ). These operators are a particular advantage of a matrix programming language.

Additional classes of operators are detailed in [M-2] **Intro**.

### B.2.3 Mata functions

Standard matrix functions have arguments provided in parentheses,  $()$ .

## Scalar and matrix functions

Some matrix commands produce scalars, for example,

```
: // Scalar functions of a matrix
: r = rows(A)
: r
2
```

Commonly used examples include those for matrix determinant (`det()`), rank (`rank()`), and trace (`trace()`). Statistical functions include `mean()`.

Some matrix commands produce matrices by element-by-element transformation, for example,

```
: // Matrix function that returns matrix by element-by-element transformation
: D = sqrt(A)
: D
      1           2           3
1 | 1   1.414213562  1.732050808
2 | 2   2.236067977  2.449489743
```

Mathematical functions include absolute value (`abs()`), sign (`sign()`), natural logarithm (`ln()`), exponentiation (`exp()`), log factorial (`lnfactorial()`), modulus (`mod()`), and truncation to integer (`trunc()`). Among the statistical functions are uniform draws (`runiform()`), standard normal density (`normal()`), and many other densities and cumulative distribution functions.

Some matrix commands produce vectors and matrices by acting on the whole matrix. A leading example is matrix inversion, discussed below. The `mean()` function finds the mean of columns of a matrix, and `corr()` forms a correlation matrix from a variance matrix.

Eigenvalues and eigenvectors of a square matrix can be obtained by using the Mata `eigensystem()` function, for example,

```

: // Calculate eigenvalues and eigenvectors
: E = (1, 2 \ 4, 3)
: lamda = .
: eigvecs = .
: eigensystem(E,eigvecs,lambda)
: lambda
      1      2
1 5 -1
: eigvecs
      1      2
1 -.447213595 -.707106781
2 -.894427191 .707106781

```

The eigenvalues are in the row vector `lambda`, and the eigenvectors are the corresponding columns of the square matrix `eigvecs`. The command requires that `lambda` and `eigvecs` already exist, so we initialized them as missing values.

Mata has many functions; see [M-4] **Intro** for an index and guide to functions.

### Matrix inversion

There are several different matrix inversion functions. The `cholinv()` function, the fastest, computes the inverse of a positive-definite symmetric matrix. The `invsym()` function computes the inverse of a real symmetric matrix; `luinv()` computes the inverse of a square matrix; `qrinv()` computes the generalized inverse of a matrix; and `pinv()` computes the Moore-Penrose pseudoinverse.

For the full-column rank matrix `x`, the matrix `x' x` is positive-definite symmetric, so `cholinv(x' x)` is best. But this function will fail if `x' x` is not precisely symmetric because of a rounding error in calculations. The `makesymmetric()` function forms a symmetric matrix by copying elements below the diagonal into the corresponding position above the diagonal, for example,

```

: // Use of makesymmetric() before cholinv()
: F = 0.5*I(2)
: G = makesymmetric(cholinv(F`F))
: E
[symmetric]
    1      2
  1  4
  2      4

```

## B.2.4 Mata cross products

The matrix `cross()` function creates matrix cross products. For example, `cross(X,X)` forms  $X'X$ , `cross(X,Z)` forms  $X'Z$ , and `cross(X,w,Z)` forms  $X'\text{diag}(w)Z$ . For the data loaded earlier into `x` and `y`, the OLS estimator can be computed as

```

: // Matrix cross product
: beta = (cholinv(cross(X,X)))*(cross(X,y))
: beta
    1
  1  .1353097647
  2  4.468739434

```

These estimates equal those given in appendix [A.1.2](#).

The advantages of using `cross()`, rather than the arithmetic multiplication operator, are faster computation and less memory use. Rows with missing observations are dropped, whereas  $X'Z$  will produce missing values everywhere if there are any missing observations. And `cross(X'X)` produces a symmetric result so that there is no longer a need to use the `makesymmetric()` function before `cholinv()` or `invsym()`.

## B.2.5 Mata matrix subscripts and combining matrices

The  $(i,j)$ th entry in a matrix is denoted by `[i,j]`. For example, to set the  $(1,2)$  entry in matrix `A` to equal the  $(1,1)$  entry, type the command

```
: // Matrix subscripts  
: A[1,2] = A[1,1]
```

```
: A
```

1	2	3
---	---	---

1	1	3
2	4	6

For a column vector, the  $i$ th entry is denoted by `[i, 1]` rather than simply `[i]`. Similarly, for a row vector, the  $j$ th entry is denoted by `[1, j]` rather than simply `[j]`.

To add columns to a matrix, use the horizontal concatenation operator, a comma. Thus, `A, B` adds the columns of `B` after the columns of `A`, assuming the two matrices have the same number of rows. For example, type

```
: // Combining matrices: add columns  
: M = A, A
```

```
: M
```

1	2	3	4	5	6
---	---	---	---	---	---

1	1	3	1	1	3
2	4	5	6	4	6

To add or append rows to a matrix, use the vertical concatenation operator, a backslash. Thus, `A \ B` adds the rows of `B` after the rows of `A`, assuming the two matrices have the same number of columns. For example, type

```
: // Combining matrices: add rows  
: N = A \ A
```

```
: N
```

1	2	3
---	---	---

1	1	3
2	4	6
3	1	3
4	4	6

A submatrix can be extracted from a matrix by using list subscripts that give as a first argument the rows being extracted and as a second argument

the columns being extracted. For example, to extract the submatrix formed by rows 1–2 and columns 5–6 of the matrix  $M$ , we type

```
: // Form submatrix using list subscripts
: O = M[(1\2), (5::6)]
: O
      1   2

$$\begin{array}{|c|c|} \hline 1 & 1 & 3 \\ \hline 2 & 5 & 6 \\ \hline \end{array}$$

```

An alternative is to use range subscripts that give the subscripts for the upper-left entry and the lower-right entry of the portion to be extracted. Thus, type

```
: // Form submatrix using range subscripts
: P = M[|1,5 \ 2,6|]
: P
      1   2

$$\begin{array}{|c|c|} \hline 1 & 1 & 3 \\ \hline 2 & 5 & 6 \\ \hline \end{array}$$

```

Where both list and range subscripts can be used, range subscripts are preferred because they execute quicker. For more details, see [M-2] **Subscripts**.

## B.2.6 Transferring Mata data and matrices to Stata

Mata functions beginning with `st_` provide an interface with Stata.

### Creating Stata matrices from Mata matrices

A Stata matrix can be created from a Mata matrix by using the Mata `st_matrix()` function.

For example, to create a Stata matrix,  $Q$ , from the Mata matrix  $P$  and then list the Stata matrix, type

```

: // Output Mata matrix to Stata
: st_matrix("Q", P)
: stata("matrix list Q")
Q[2,2]
    c1  c2
r1   1   3
r2   5   6

```

Section 3.9 provides an example where the parameter vector `b` and the estimate of the variance–covariance matrix of the estimator are computed in Stata and passed from Mata to Stata with `st_matrix()`, and then results are posted and nicely displayed by using the Stata `ereturn` command.

### Creating Stata data from a Mata vector

The `st_addvar()` function adds a new variable to a Stata dataset, though it creates only the name of the variable and not its values. The `st_store()` function modifies the values of a variable currently in a Stata dataset. Thus, to create a new variable in Stata and give that new variable values, we type `st_addvar()` followed by `st_store()`.

Recall that `x` is a matrix with the variables `totchr` and `cons` and that `beta` is a column vector of OLS coefficients from the regression of `ltotexp` on `totchr` and `cons`. We create the vector of fitted values, `yhat`, in Mata, pass these to Stata as the `ltotexphat` variable, and use the `summarize` command to check the results. We have

```

: // Output Mata matrix to Stata
: yhat = X*beta
: st_addvar("float", "ltotexphat")
46
: st_store(., "ltotexphat", yhat)

: stata("summarize ltotexp ltotexphat")

```

Variable	Obs	Mean	Std. dev.	Min	Max
<code>ltotexp</code>	100	4.533688	.8226942	1.098612	5.332719
<code>ltotexphat</code>	100	4.533688	.0970792	4.46874	4.874669

As expected after OLS regression, the average of the fitted values equals the average of the dependent variable.

## B.3 Programming in Mata

Detailed examples using Mata code are presented in sections [3.9](#), 16.2, 16.8, 30.3, and 30.4, and appendix [C.2](#). These examples pass data from Stata to Mata, calculate parameter estimates and estimates of the variance–covariance matrix of the estimator in Mata, and pass these back to Stata.

Here we present a very introductory treatment of programming in Mata.

### B.3.1 Mata program

As an example, we create a Mata program, `calcsum`, that calculates the column sum of a column vector. This example, based on the example in [M-1] **Ado**, is purely illustrative because the Mata `colsum()` function does this anyway.

The column vector `x` is obtained from a variable named `varname` in the Stata dataset currently in memory by using the `st_view()` function. The `varname` string is a program argument supplied when the program is called. The actual calculation of the column sum is done with the Mata `colsum()` function. The result is put in the real scalar `resultissum`, a second program argument. To apply the program to the `ltotexp` variable, we call the `calcsum` program with the "`ltotexp`" and `sum` arguments. The result is in `sum`, and to see the result, we simply type `sum`. We have

```
. mata:  
: void calcsum(varname, resultissum) _____ mata (type end to exit)  
> {  
>     st_view(x=., ., varname)  
>     resultissum = colsum(x)  
> }  
: sum = .  
: calcsum("ltotexp", sum)  
: sum  
453.3688121  
: end
```

The result, 453.3688, is that expected because the sample mean of the 100 observations on `ltotexp` was 4.533688 from output given in appendix [B.2.6](#).

### B.3.2 Mata program with results output to Stata

The preceding Mata program passes the result, `resultissum`, back to Mata. We next consider a variation that passes the result, renamed `sum`, to Stata.

To transfer the result to Stata, we use the Mata `st_numscalar()` function and drop the second argument in the `calcsum` program because the result is no longer passed to Mata. Because the result is now in Stata, we need to use the Stata `display` command to display the result. We have

```
. mata:  
----- mata (type end to exit) -----  
: void function calcsum2(varname)  
> {  
>     st_view(x=., ., varname)  
>     st_numscalar("r(sum)", colsum(x))  
> }  
:  
: calcsum2("ltotexp")  
:  
: stata("display r(sum)")  
453.36881  
:  
: end
```

---

### B.3.3 Stata program that calls a Mata program

The preceding two programs call the Mata program from within Mata. We now create a Stata program, `varsum`, that calls the Mata program `calcsum2` from Stata.

The `varsum` program uses standard Stata syntax (see appendix [A.2.7](#)) rather than positional arguments. This syntax recognizes the argument in the call `varsum ltotexp` as a variable name that is placed in `varlist`. The Mata program `calcsum2`, already defined in the preceding section, is called with `varname` being the variable name in `varlist`. We have

```
. program varsum
1.      version 17
2.      syntax varname
3.      mata: calcsum2(``varlist'')
4.      display r(sum)
5. end
. varsum ltotexp
453.36881
```

### B.3.4 Using Mata in ado-files

The main construct for writing new commands in Stata is a Stata ado-file. When computation in Mata is convenient, the ado-file can include Mata code or call a Mata function.

A Mata function defined in an ado-file requires compilation every time it is called. To save computer time, one can reuse compiled functions without the need for recompilation by using the `mata mosave` and `mata mlib` commands. For details, see [M-1] **Ado**, which presents the preceding column sum example in much more generality.

### B.3.5 Declarations

The examples in section 16.8 and appendix [C](#) include a Mata program with arguments to be passed to and from the Mata `optimize()` function.

The code in these examples does not declare matrices and scalars ahead of their use. This makes coding easier but makes it more likely that errors may go undetected. For example, if an operation is expected to create a scalar but a matrix is the result, there may be no message to this effect. If instead we had previously declared the expected result to be a scalar, then an error would occur if a matrix was erroneously created.

The following Mata code rewrites the `optimize()` function evaluator `pmlegf2` program in appendix [C.2.3](#) to declare the types of all program arguments and all other variables used in the program.

```
: void pmlegf2(real scalar todo,
>             real rowvector b,
>             real colvector y,
>             real matrix X,
>             real colvector lndensity,
>             real matrix g,
>             real matrix H)
> {
>     real colvector Xb
>     real colvector mu
>     Xb = X*b'
>     mu = exp(Xb)
>     lndensity = -mu + y:*Xb - lnfactorial(y)
>     if (todo == 0) return
>     g = (y-mu):*X
>     if (todo == 1) return
>     H = - cross(X, mu, X)
> }
```

The Mata command `mata set matastrict on` requires that declarations be provided.

## B.4 Additional resources

The [M] *Stata Mata Reference Manual* provides all the Mata commands. [Gould \(2018\)](#) provides an exposition of Mata, and [Baum \(2016\)](#) includes coverage of Mata.

## Appendix C

# Optimization in Mata

The Stata `ml` command is useful for objective functions of form  $Q(\boldsymbol{\theta}) = \sum_{i=1}^N q(\mathbf{y}_i, \mathbf{x}_i, \boldsymbol{\theta})$ . Not all optimization problems fall into this class, however, in which case one needs to use the Mata `moptimize()` function or the more flexible Mata `optimize()` function.

This appendix presents these Mata optimization functions, building on the optimization methods given in chapter 16 and Mata commands summarized in appendix [B](#).

Note that if the simpler Stata `ml` command can be used, then there is no benefit in using the `moptimize()` function because the `ml` command is just a front end to the `moptimize()` function. The `moptimize()` function in turn uses some features of the `optimize()` function.

Applications in this appendix use the same data as in chapter 16.

```
. * Read in data
. use mus210mepsdocvisyoung, clear
(A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e)
. qui keep if year02 == 1
```

## C.1 Mata `moptimize()` function

The Mata `moptimize()` function is especially useful for models with  $q(\mathbf{y}_i, \mathbf{x}_i, \boldsymbol{\theta})$  of single-index form  $q(\mathbf{y}_i, \mathbf{x}'_i \boldsymbol{\theta})$  or of multi-index form  $q(\mathbf{y}_i, \mathbf{x}'_{1i} \boldsymbol{\theta}_1, \dots, \mathbf{x}'_{J_i} \boldsymbol{\theta}_J)$ .

### C.1.1 `moptimize()` evaluators `lf`, `d`, `gf`, and `q`

An evaluator function is one that provides the formula to calculate the function being maximized and defines the vector that is the maximand. It can additionally provide the formula for first and second derivatives and define any data on a dependent variable and regressors that determine the value of the objective function  $Q(\boldsymbol{\theta})$ .

There are several distinct types of evaluator functions used by the `moptimize()` function. For all evaluators, the first argument is  $M$ , which is a handle, initially obtained from `m_optimize_init()`, that can then be passed as an argument to other `moptimize()` functions.

Detailed examples of evaluators are given below. Most evaluators have syntax

$$\text{evaluator}(M, \text{todo}, b, fv, S, H)$$

where  $M$  is a handle,  $\text{todo}$  is a real scalar equal to 0, 1, or 2 depending on whether a gradient and Hessian are provided,  $b$  is the coefficient vector,  $f_v$  defines the objective function,  $S$  is a gradient vector or matrix, and  $H$  is a Hessian matrix.

### C.1.2 `moptimize()` functions

The `moptimize()` functions fall into four broad categories. Functions that define the optimization problem, such as the name of the evaluator and the iterative technique to be used, begin with `moptimize_init`. Functions that

lead to optimization are `moptimize()` or `moptimize_evaluate()`. Functions that return results begin with `moptimize_return`. And function `moptimize_query()` lists optimization settings and results.

A complete listing of these functions is given in [M-5] **`moptimize()`**. The following examples illustrate various `moptimize()` functions to perform Poisson regression of  $y$  on  $x$  and obtain coefficient estimates and an estimate of the associated variance–covariance matrix of the estimator. This simple example has only one dependent variable and only one index.

### C.1.3 `moptimize()` methods **`lf`, `lf0`, `lf1`, and `lf2`**

The type `lf` evaluator defines the individual contributions  $q_i(\theta)$  to the objective function and has syntax

$$\text{evaluator}(M, b, fv)$$

where  $b$  is the  $1 \times K$  coefficient vector and  $f v$  is an  $N \times 1$  column vector with each observation's contribution to the objective function  $Q(\mathbf{b})$ ;  $Q(\mathbf{b})$  is then the column sum of  $\mathbf{b}$ .

The following program defines an `lf` evaluator for `moptimize()`. The first line `y1 = moptimize_util_depvar(M, 1)` returns a column vector  $y1$  containing the values of the dependent variable whose values are set later by `moptimize_init_depvar(M, 1, " ")`. The second line `xb = moptimize_util_xb(M, b, 1)` returns a column vector containing the values of the index  $\mathbf{x}'\theta$ , where the regressors are defined in the `moptimize_init_eq_indepvars(M, 1, " ")` program below. The third line provides a column vector with the log density for each observation, that is, for each observation's contribution to the objective function.

```
. * moptimize() method lf: Program poissonlf gives lnf(y_i)
. mata:
----- mata (type end to exit) -----
: mata clear
: function poissonlf(transmorphic M, real rowvector b, fv)
> {
>     y1 = moptimize_util_depvar(M, 1)    // N x 1 vector y
>     Xb = moptimize_util_xb(M, b, 1)    // N x 1 vector Xb
>     fv = -exp(Xb):+ (y1:*Xb) :- lnfactorial(y1)
> }
: end
```

---

Given that definition of the evaluator, we move to optimization. The first line is always needed and defines `M` as the handle (we could give some other letter here). The second line specifies to use program `poissonlf` as the evaluator. The third line says that this is an `lf` evaluator. The fourth line defines the single dependent variable, and the fifth line defines the variables used to form the single index for this example. The `moptimize(M)` function initiates the optimization and `moptimize_result_display(M)` function reports the results with default standard errors.

```

. * moptimize() method lf: Implement with default standard errors
. mata:
                                         mata (type end to exit) -----
:   M = moptimize_init()
:   moptimize_init_evaluator(M, &poissonlf())
:   moptimize_init_evaluatortype(M, "lf")
:   moptimize_init_depvar(M, 1, "docvis")
:   moptimize_init_eq_indepvars(M, 1, "private chronic female income")
:   moptimize(M)
initial:      f(p) = -33899.609
alternative:   f(p) = -28031.767
rescale:      f(p) = -24020.669
Iteration 0:   f(p) = -24020.669
Iteration 1:   f(p) = -24010.663
Iteration 2:   f(p) = -18541.285
Iteration 3:   f(p) = -18503.604
Iteration 4:   f(p) = -18503.549
Iteration 5:   f(p) = -18503.549
:   moptimize_result_display(M)           // Default standard errors
                                         Number of obs = 4,412

```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.7986654	.027719	28.81	0.000	.7443372 .8529936
chronic	1.091865	.0157985	69.11	0.000	1.060901 1.12283
female	.4925481	.0160073	30.77	0.000	.4611744 .5239218
income	.003557	.0002412	14.75	0.000	.0030844 .0040297
_cons	-.2297263	.0287022	-8.00	0.000	-.2859816 -.1734711

---

```
: end
```

---

The results are identical to those obtained using the `poisson` command with default standard errors.

The function `moptimize_result_display(M, "robust")` reports the results with heteroskedastic-robust standard errors.

```

. * moptimize() method lf: Implement with heteroskedastic-robust standard errors
. mata:
:                                         mata (type end to exit) -----
:     moptimize_result_display(M, "robust") // Heteroskedastic-robust
> standard errors

```

Number of obs = 4,412

docvis	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
private	.7986654	.1090015	7.33	0.000	.5850265	1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167	1.201614
female	.4925481	.0585365	8.41	0.000	.3778187	.6072775
income	.003557	.0010825	3.29	0.001	.0014354	.0056787
_cons	-.2297263	.1108733	-2.07	0.038	-.4470339	-.0124188

```
: end
```

To obtain cluster-robust standard errors, we first need to define the cluster identification variable, here `age`, using the `moptimize_init_cluster(M, " ")` function. The `moptimize_result_display(M)` function then reports cluster-robust standard errors.

```

. * moptimize() method lf: Implement with cluster--robust standard errors
. mata:
:                                         mata (type end to exit) -----
:     M = moptimize_init()
:     moptimize_init_evaluator(M, &poissonlf())
:     moptimize_init_evaluatortype(M, "lf")
:     moptimize_init_depvar(M, 1, "docvis")
:     moptimize_init_eq_indepvars(M, 1, "private chronic female income")
:     moptimize_init_cluster(M, "age") // Define the cluster variable
:     moptimize(M)
initial:    f(p) = -33899.609
alternative: f(p) = -28031.767
rescale:    f(p) = -24020.669
Iteration 0: f(p) = -24020.669
Iteration 1: f(p) = -24010.663
Iteration 2: f(p) = -18541.285
Iteration 3: f(p) = -18503.604
Iteration 4: f(p) = -18503.549
Iteration 5: f(p) = -18503.549

```

```

:      moptimize_result_display(M)      // Now gives cluster--robust standard errors
                                                Number of obs = 4,412
                                                (Std. err. adjusted for 40 clusters in age)

```

---

docvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
private	.7986654	.1496492	5.34	0.000	.5053583 1.091972
chronic	1.091865	.0603102	18.10	0.000	.9736593 1.210071
female	.4925481	.0686028	7.18	0.000	.3580891 .627007
income	.003557	.0011792	3.02	0.003	.0012458 .0058683
_cons	-.2297263	.1453959	-1.58	0.114	-.514697 .0552443

---

```
: end
```

---

The type `lf*` evaluators additionally allow for specification of first and second derivatives at the individual observation level. The syntax is

$$\text{evaluator}(M, \text{todo}, b, fv, S, H)$$

where `todo` is a real scalar equal to 0, 1, or 2 (for, respectively, `lf0`, `lf1`, and `lf2`); `b` is the  $1 \times K$  coefficient vector; `fv` is again an  $N \times 1$  column vector with each observation's contribution to the objective function; `S` is an  $N \times K$  matrix of scores, whose column sum gives the gradient vector; and `H` is a  $K \times K$  Hessian matrix.

We next consider an `lf2` evaluator for `moptimize()` that provides an  $N \times K$  matrix of first derivatives, where each row is for a single observation, and a  $K \times K$  matrix of second derivatives of the objective function. This latter matrix requires summing over observations, which is done using the `moptimize_util_matsum()` function.

```
. * moptimize() method lf2: Add first and second derivatives to poissonlf
. mata:
----- mata (type end to exit) -----
: mata clear
: function poissonlf2(transmorphic M, real scalar todo,
> real rowvector b, fv, S, H)
> {
>     y1 = moptimize_util_depvar(M, 1)      // N x 1 vector y
>     Xb = moptimize_util_xb(M, b, 1)        // N x 1 vector Xb
>     fv = -exp(Xb):+ (y1:*Xb) :- lnfactorial(y1)
>     if (todo>=1) {
>         s1 = (y1:-exp(Xb))
>         S = s1
>         if (todo==2) {
>             h11 = -exp(Xb)
>             H11 = moptimize_util_matsum(M, 1,1, h11, 0)
>             H = H11
>         }
>     }
> }
> }
: end
```

---

Optimization then yields

```

. * moptimize() method lf2: Implement with heteroskedastic-robust standard errors
. mata:
                                         mata (type end to exit) -----
: M = moptimize_init()
: moptimize_init_evaluator(M, &poissonlf2())
: moptimize_init_evaluator_type(M, "lf2")
: moptimize_init_depvar(M, 1, "docvis")
: moptimize_init_eq_indepvars(M, 1, "private chronic female income")
: moptimize(M)
initial:      f(p) = -33899.609
alternative:   f(p) = -28031.767
rescale:       f(p) = -24020.669
Iteration 0:   f(p) = -24020.669
Iteration 1:   f(p) = -18845.463
Iteration 2:   f(p) = -18510.192
Iteration 3:   f(p) = -18503.551
Iteration 4:   f(p) = -18503.549
Iteration 5:   f(p) = -18503.549
: moptimize_result_display(M, "robust") // Robust standard errors

```

Number of obs = 4,412

docvis	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
private	.7986654	.1090015	7.33	0.000	.5850265	1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167	1.201614
female	.4925481	.0585365	8.41	0.000	.3778187	.6072775
income	.003557	.0010825	3.29	0.001	.0014354	.0056787
_cons	-.2297263	.1108733	-2.07	0.038	-.4470339	-.0124187

: end

### C.1.4 moptimize() methods d0, d1, and d2

The type `d*` evaluators directly define the objective function  $Q(\theta)$ , rather than the individual contributions  $q_i(\theta)$ . The syntax is

$$\text{evaluator}(M, \text{todo}, b, fv, S, H)$$

where `todo` is a real scalar equal to 0, 1, or 2 (for, respectively, `d0`, `d1`, and `d2`);  $b$  is the  $1 \times K$  coefficient vector;  $fV$  is the scalar  $Q(b)$ ;  $S$  is a  $1 \times K$  gradient vector; and  $H$  is a  $K \times K$  Hessian matrix.

The following example provides a `d2` evaluator. Now the first derivatives are with respect to the entire objective function, yielding a  $K \times 1$  column vector that is constructed from individual observations using the `moptimize_util_vecsum()` function. The  $K \times K$  matrix of second derivatives is calculated in the same way as for the `lf2` evaluator.

```
. * moptimize() method d2: Program poissond2
. mata:

$$\begin{aligned} & \text{mata clear} \\ & \text{function poissond2(transmorphic M, real scalar todo,} \\ > & \quad \text{real rowvector b, fv, g, H)} \\ > & \{ \\ > & \quad y1 = moptimize_util_depvar(M, 1) // N x 1 vector y1 \\ > & \quad Xb = moptimize_util_xb(M, b, 1) // N x 1 vector Xb \\ > & \quad fv = moptimize_util_sum(M, -exp(Xb) :+ (y1:*Xb) :- lnfactorial(y1)) \\ > & \quad \text{if (todo}>=1) \{ \\ > & \quad \quad s1 = (y1:-exp(Xb)) \\ > & \quad \quad g1 = moptimize_util_vecsum(M, 1, s1, fv) \\ > & \quad \quad g = g1 \\ > & \quad \quad \text{if (todo==2) \{ \\ > & \quad \quad \quad h11 = -exp(Xb) \\ > & \quad \quad \quad H11 = moptimize_util_matsum(M, 1,1, h11, fv) \\ > & \quad \quad \quad H = H11 \\ > & \quad \quad \}} \\ > & \quad \}} \\ > & \} \\ : & \text{end} \end{aligned}$$

```

---

Optimization then yields

```

. * moptimize() method d2: Can implement only with default standard errors
. mata:
                                         mata (type end to exit) -----
:   M = moptimize_init()
:   moptimize_init_evaluator(M, &poissonD2())
:   moptimize_init_evaluatortype(M, "d2")
:   moptimize_init_depvar(M, 1, "docvis")
:   moptimize_init_eq_indepvars(M, 1, "private chronic female income")
:   moptimize(M)
initial:      f(p) = -33899.609
alternative:   f(p) = -28031.767
rescale:       f(p) = -24020.669
Iteration 0:   f(p) = -24020.669
Iteration 1:   f(p) = -18845.463
Iteration 2:   f(p) = -18510.192
Iteration 3:   f(p) = -18503.551
Iteration 4:   f(p) = -18503.549
Iteration 5:   f(p) = -18503.549

:   moptimize_result_display(M)
                                         Number of obs = 4,412


```

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.7986654	.027719	28.81	0.000	.7443372 .8529936
chronic	1.091865	.0157985	69.11	0.000	1.060901 1.12283
female	.4925481	.0160073	30.77	0.000	.4611744 .5239218
income	.003557	.0002412	14.75	0.000	.0030844 .0040297
_cons	-.2297263	.0287022	-8.00	0.000	-.2859816 -.1734711

---

```
: end
```

The resulting optimization can compute only default standard errors because robust standard errors require the first derivative for each observation, whereas program `poissonD2` provided only the first derivative of the entire objective function.

```
. * moptimize() method d2: Cannot obtain heteroskedastic-robust standard errors
. mata:
```

```
                                mata (type end to exit) -----
:      moptimize_result_display(M, "robust")
```

Number of obs = 4,412

docvis	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
private	.7986654	.	.	.	.	.
chronic	1.091865	.	.	.	.	.
female	.4925481	.	.	.	.	.
income	.003557	.	.	.	.	.
_cons	-.2297263	.	.	.	.	.

```
: end
```

The following code manually computes heteroskedastic–robust standard errors given the preceding parameter estimates.

```

. * moptimize() method d2: Compute heteroskedastic-robust standard errors manually
. capture generate cons == 1           // We need to add a constant here
. mata:
   mata (type end to exit)
:   b = moptimize_result_coefs(M)    // Estimates from previous moptimize()
:   st_view(X=., ., tokens("private chronic female income cons"))
:   st_view(y=., ., tokens("docvis"))
:   N = rows(X)
:   Xb = X*b'
:   mu = exp(Xb)
:   XmuXinv = cholinv(cross(X,mu,X))
:   residsq = (y-mu):*(y-mu)
:   Vrobust = (N/(N-1))*XmuXinv`*(cross(X,residsq,X))*XmuXinv
:   st_matrix("b",b)                // Pass results from Mata to Stata
:   st_matrix("V",Vrobust)         // Pass results from Mata to Stata
: end
.
. matrix colnames b = private chronic female income cons
. matrix colnames V = private chronic female income cons
. matrix rownames V = private chronic female income cons
. ereturn post b V
. ereturn display

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.7986654	.1090015	7.33	0.000	.5850265 1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821167 1.201614
female	.4925481	.0585365	8.41	0.000	.3778187 .6072775
income	.003557	.0010825	3.29	0.001	.0014354 .0056787
cons	-.2297263	.1108733	-2.07	0.038	-.4470339 -.0124187

### C.1.5 moptimize() methods gf0, gf1, and gf2

The type `gf` evaluators are a variation of type `lf*` evaluators that permit the objective function to be of form  $\sum_{i=1}^N \mathbf{q}_i(\boldsymbol{\theta})$ , where  $\mathbf{q}_i(\cdot)$  is an  $L \times 1$  vector rather than a scalar. This is especially useful for fitting panel-data models. The syntax is

$$\text{evaluator}(M, \text{todo}, b, fv, S, H)$$

where *todo* is a real scalar equal to 0, 1, or 2 (for, respectively, `gf0`, `gf1`, and `gf2`);  $b$  is the  $1 \times K$  coefficient vector;  $fv$  is an  $L \times 1$  vector, where  $L$  is the number of independent elements;  $S$  is an  $L \times K$  gradient vector; and  $H$  is a  $K \times K$  Hessian matrix.

For the current cross-sectional example, there is no need to use a `gf*` evaluator because  $q_i(\cdot)$  is scalar.

### C.1.6 moptimize() methods q0 and q1

`q0` and `q1` evaluators are for an objective function of the quadratic form  $\mathbf{h}(\boldsymbol{\theta})' \mathbf{W} \mathbf{h}(\boldsymbol{\theta})$ , as is the case for minimum chi-squared estimators. In this case, the objective function is minimized.

The type `q` evaluators are used in the special case of a quadratic objective function  $Q(\mathbf{b}) = \mathbf{r}(\mathbf{b})' \mathbf{W} \mathbf{r}(\mathbf{b})$  and have syntax

$$\text{evaluator}(M, \text{todo}, b, r, S)$$

where *todo* is a real scalar equal to 0 or 1 (for, respectively, `q0` or `q1`);  $b$  is the  $1 \times K$  coefficient vector;  $r$  is an  $L \times 1$  vector, where  $L$  is the number of independent elements; and  $S$  is an  $L \times K$  gradient vector.

As an example, we consider the Poisson maximum-likelihood estimator (MLE), which has first-order conditions  $\sum_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \mathbf{x}_i = \mathbf{0}$ . This is equivalent to maximizing the quadratic form  $\mathbf{h}(\boldsymbol{\beta})' \mathbf{h}(\boldsymbol{\beta})$ , where  $\mathbf{h}(\boldsymbol{\beta}) = \sum_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \mathbf{x}_i$ . Here  $L = K$  and  $\mathbf{W}$  is the  $K \times K$  identity matrix.

The `q0` evaluator function for this example is

```

. * moptimize() poisson q0: Poisson example
. capture generate cons == 1           // We need to add a constant here
. mata:
                                         mata (type end to exit) -----
: mata clear
: function poissonq0(transmorphic M, real scalar todo,
>     real rowvector b, r, S)
> {
>     y = moptimize_util_depvar(M, 1)      // N x 1 vector y
>     Xb = moptimize_util_xb(M, b, 1) // N x 1 vector Xb
>     st_view(X=., ., tokens("private chronic female income cons"))
>     mu = exp(Xb)                      // N x 1 vector mu
>     r = X`*(y-mu)                    // k x 1 vector r
> }
note: argument todo unused.
note: argument S unused.
: end

```

---

The `moptimize_init_gnweightmatrix( $M, W$ )` function specifies the  $L \times L$  fixed-weighting matrix  $\mathbf{W}$ . If this is not specified, then it is assumed that  $\mathbf{W} = \mathbf{I}$ , the case here. Optimization yields

```

. * moptimize() method gf1: Implement with heteroskedastic-robust
> standard errors
. mata:
                                         mata (type end to exit) -----
:     M = moptimize_init()
:     moptimize_init_evaluator_type(M, "q0")
:     moptimize_init_evaluator(M, &poissonq0())
:     moptimize_init_vcetype(M, "robust")
:     moptimize_init_depvar(M, 1, "docvis")
:     moptimize_init_eq_indepvars(M, 1, "private chronic female income")
:     moptimize_init_technique(M,"gn")
:     moptimize_init_conv_maxiter(M, 10)
:     moptimize(M)
initial:      f(p) = 2.814e+11
alternative:   f(p) = 1.867e+11
rescale:       f(p) = 7.285e+10
Iteration 0:   f(p) = 7.285e+10
Iteration 1:   f(p) = 6.841e+08
Iteration 2:   f(p) = 91618347
Iteration 3:   f(p) = 24112.076
Iteration 4:   f(p) = .06168098
Iteration 5:   f(p) = 3.938e-12
Iteration 6:   f(p) = 5.332e-19
:     b = moptimize_result_coefs(M)
:     b
          1           2           3           4           5
1  .7986653786    1.091865108    .4925480693    .0035570127   -.2297263379
: end

```

The estimated coefficients are the same as those from `poisson`. Note that function `poissonq0()` had to include a constant in `X`, while `moptimize_init_eq_indepvars()` automatically included the constant.

## C.2 Mata optimize() function

The Mata `optimize()` function can handle more general models than the `moptimize()` function. A simple Poisson example is given here. A more complex overidentified nonlinear generalized method of moments example is presented in section 16.8.

### C.2.1 optimize() d and gf evaluators

Because  $y$  and  $x$  are used to denote dependent variables and regressors, the Mata documentation uses the generic notation that we want to compute real row vector  $\mathbf{p}$  that maximizes the scalar function  $f(\mathbf{p})$ . Note that  $\mathbf{p}$  is a row vector, whereas in this book, we usually define vectors (such as  $\boldsymbol{\beta}$ ) to be column vectors.

An evaluator function calculates the value of the objective function at values of the parameter vector. It may optionally calculate the gradient and the Hessian.

There are two distinct types of evaluator functions used by Mata.

A type `d` evaluator returns the value of the objective as the scalar  $v = f(\mathbf{p})$ . The minimal syntax is

```
void evaluator(todo, p, v, g, H)
```

where  $todo$  is a scalar,  $p$  is the row vector of parameters,  $v$  is the scalar function value,  $g$  is the gradient row vector  $\partial f(\mathbf{p})/\partial \mathbf{p}$ , and  $H$  is the Hessian matrix  $\partial^2 f(\mathbf{p})/\partial \mathbf{p} \partial \mathbf{p}'$ . If  $todo$  equals zero, then numerical derivatives are used (method `d0`), and  $g$  and  $H$  need not be provided. If  $todo$  equals one, then  $g$  must be provided (method `d1`), and if  $todo$  equals two, then both  $g$  and  $H$  must be provided (method `d2`).

A type `gf` evaluator is more suited to  $m$ -estimation problems, where we maximize  $Q(\boldsymbol{\theta}) = \sum_{i=1}^N q_i(\boldsymbol{\theta})$ . Then it may be more convenient to provide an  $N \times 1$  vector with the  $i$ th entry  $q_i(\boldsymbol{\theta})$  rather than the scalar  $Q(\boldsymbol{\theta})$ . A type

`gf` evaluator returns the column vector  $\mathbf{v}$ , and  $f(\mathbf{p})$  equals the sum of the entries in  $\mathbf{v}$ . The minimal syntax for a type `gf` evaluator is

```
void evaluator(todo, p, v, g, H)
```

where  $todo$  is a scalar,  $p$  is a row vector of parameters,  $v$  is a column vector,  $g$  is now the gradient matrix  $\partial\mathbf{v}/\partial\mathbf{p}$ , and  $H$  is the Hessian matrix. If  $todo$  equals zero, then numerical derivatives are used (method `gf0`), and  $g$  and  $H$  need not be provided. If  $todo$  equals one, then  $g$  must be provided (method `gf1`), and if  $todo$  equals two, then both  $g$  and  $H$  must be provided (method `gf2`).

Up to nine additional arguments can be provided in these evaluators, appearing after  $p$  and before  $v$ . In that case, these arguments and their relative positions need to be declared by using the `optimize_init_arguments()` function, illustrated below. For regression with data in  $\mathbf{y}$  and  $\mathbf{X}$ , the arguments will include  $y$  and  $X$ .

## C.2.2 Optimize functions

The `optimize()` functions fall into four broad categories, similar to those already discussed for the `moptimize()` function. First, functions that define the optimization problem, such as the name of the evaluator and the iterative technique to be used, begin with `optimize_init`. Second, functions that lead to optimization are `optimize()` or `optimize_evaluate()`. Third, functions that return results begin with `optimize_result`. Fourth, the `optimize_query()` function lists optimization settings and results.

A complete listing of these functions and their syntaxes is given in [M-5] **optimize()**. The following example essentially uses the minimal set of `optimize()` functions to perform a (nonlinear) regression of  $y$  on  $\mathbf{x}$  and to obtain coefficient estimates and an estimate of the associated variance-covariance matrix of the estimator.

## C.2.3 Poisson example

We implement the Poisson MLE, using the Mata `optimize()` function method `gf2`.

## Evaluator program for Poisson MLE

The key ingredient is the evaluator program, named `pmlegf2()`. Because the `gf2` method is used, the evaluator program needs to evaluate a vector of log densities, named `lndensity`, an associated gradient matrix, named `g`, and the Hessian, named `H`. We name the parameter vector `b`. The dependent variable and the regressor matrix, named `y` and `x`, respectively, are two additional program arguments that will need to be declared by using the `optimize_init_argument()` function.

For the Poisson MLE, from section 16.2.2, the column vector of log densities has the  $i$ th entry  $\ln f(y_i | \mathbf{x}_i) = -\exp(\mathbf{x}'_i \boldsymbol{\beta}) + \mathbf{x}'_i \boldsymbol{\beta} y_i - \ln y_i!$ ; the associated gradient matrix has the  $i$ th row  $\{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \mathbf{x}_i$ ; and the Hessian is the matrix  $\sum_i -\exp(\mathbf{x}'_i \boldsymbol{\beta}) \mathbf{x}_i \mathbf{x}'_i$ . A listing of the evaluator program follows:

```
. * optimize() method gf2: Evaluator function
. mata
:   void pmlegf2(todo, b, y, X, lndensity, g, H)
>   {
>     Xb = X*b'
>     mu = exp(Xb)
>     lndensity = -mu + y:*Xb - lnfactorial(y)
>     if (todo == 0) return
>     g = (y-mu):*X
>     if (todo == 1) return
>     H = - cross(X, mu, X)
>   }
: end
```

A better version of this evaluator function that declares the types of all program arguments and other variables used in the program is given in appendix [B.3.5](#).

## The `optimize()` function for Poisson MLE

The complete Mata code has four components. First, define the evaluator, a repeat of the preceding code listing. Second, associate matrices `y` and `X` with Stata variables by using the `st_view()` function. Third, `optimize`, which at a minimum requires the seven `optimize()` functions, given below. Fourth, construct and list the key results.

```

. * optimize() method gf2: Implement with cluster--robust standard errors
. mata
                                         mata (type end to exit) -----
: mata clear
: void pmlegf2(todo, b, y, X, lndensity, g, H)
> {
>     Xb = X*b'
>     mu = exp(Xb)
>     lndensity = -mu + y:*Xb - lnfactorial(y)
>     if (todo == 0) return
>     g = (y-mu):*X
>     if (todo == 1) return
>     H = - cross(X, mu, X)
> }
: st_view(y=., ., "docvis")
: st_view(X=., ., tokens("private chronic female income cons"))
: S = optimize_init()
: optimize_init_evaluator(S, &pmlegf2())
: optimize_init_evaluatortype(S, "gf2")
: optimize_init_argument(S, 1, y)
: optimize_init_argument(S, 2, X)
: optimize_init_cluster(S, "age") // Define cluster for cluster--robust
> standard errors
: k = cols(X)
: optimize_init_params(S, J(1,k,0))

: b = optimize(S)
Iteration 0: f(p) = -33899.609
Iteration 1: f(p) = -19668.697
Iteration 2: f(p) = -18585.609
Iteration 3: f(p) = -18503.779
Iteration 4: f(p) = -18503.549
Iteration 5: f(p) = -18503.549
: Vbrob = optimize_result_V_robust(S)
: serob = (sqrt(diagonal(Vbrob)))'
: b \ serob
      1           2           3           4           5
1   .7986653788   1.091865108   .4925480693   .0035570127   -.2297263376
2   .1496492194   .0603102053   .0686027716   .0011792332   .1453958671
: end

```

The `S = optimize_init()` function initiates the optimization, and because `S` is used, the remaining functions have the first argument `S`. The next two `optimize()` functions state that the evaluator is named `pmlegf2`

and that `optimize()` method `gf2` is being used. The subsequent two `optimize()` functions indicate that the first additional argument after `b` in program `pmlegf2` is `y` and that the second is `x`. The next function provides starting values and is necessary. The `b = optimize(s)` function initiates the optimization. The remaining functions compute robust standard errors and print the results. The `optimize_init_cluster()` function defined clustering on `age`. If this line is omitted, then heteroskedastic–robust standard errors would instead be reported.

The parameter estimates and standard errors are the same as those from the Stata `poisson` command with the `vce(cluster age)` option (see section 16.5.3). Nicely displayed results can be obtained by using the `st_matrix()` function to pass `b'` and `Vbrob` from Mata to Stata and then by using the `ereturn display` command in Stata, exactly as in the section 16.2.3 example.

### C.3 Additional resources

See the [M] *Stata Mata Reference Manual* for the `moptimize()` and `optimize()` functions. [Baum \(2016\)](#), chap. 14) presents a detailed generalized method of moments example using `moptimize()`.

# Glossary of abbreviations

- **2SLS** — two-stage least squares
- **3SLS** — three-stage least squares
- **AFT** — accelerated failure time
- **AIC** — Akaike information criterion
- **AICC** — Akaike information corrected criterion
- **AIPW** — augmented inverse-probability weighting
- **AME** — average marginal effect
- **AR** — Anderson–Rubin
- **AR** — autoregressive
- **ARMA** — autoregressive moving average
- **ARUM** — additive random-utility model
- **ATE** — average treatment effect
- **ATET** — average treatment effect on the treated
- **AUC** — area under the curve
- **BC** — bias-corrected
- **BCa** — bias-corrected accelerated
- **BIC** — Bayesian information criterion
- **BLP** — Berry–Levinson–Pakes
- **CCE** — common correlated estimator
- **c.d.f.** — cumulative distribution function
- **CIF** — cumulative incidence function
- **CL** — conditional logit
- **CLR** — conditional likelihood ratio
- **CQR** — conditional quantile regression
- **CRE** — correlated random effects
- **CV** — cross-validation
- **DGP** — data-generating process
- **DIC** — deviance information criterion
- **DID** — difference in differences
- **DV** — dummy variable
- **DWH** — Durbin–Wu–Hausman
- **ERM** — extended regression model
- **ET** — endogenous treatment
- **FAQ** — frequently asked questions

- **FD** — first difference
- **FDP** — false discovery proportion
- **FDR** — false discovery rate
- **FE** — fixed effects
- **FGLS** — feasible generalized least squares
- **FMM** — finite-mixture model
- **FPC** — finite-population correction
- **FRD** — fuzzy regression discontinuity
- **FWER** — familywise error rate
- **GAM** — generalized additive models
- **GLM** — generalized linear models
- **GLS** — generalized least squares
- **GMM** — generalized method of moments
- **GS2SLS** — generalized spatial two-stage least squares
- **GSEM** — generalized structural equation model
- **GUI** — graphical user interface
- **HAC** — heteroskedasticity- and autocorrelation-consistent
- **HRS** — Health and Retirement Study
- **IIA** — independence of irrelevant alternatives
- **i.i.d.** — independent and identically distributed
- **IM** — information matrix
- **IPW** — inverse-probability weighting
- **IPW-RA** — inverse probability with regression adjustment
- **ITT** — intention to treat
- **IV** — instrumental variables
- **JIVE** — jackknife instrumental-variables estimator
- **LATE** — local average treatment effect
- **LEF** — linear exponential family
- **LIML** — limited-information maximum likelihood
- **LM** — Lagrange multiplier
- **LOOCV** — leave-one-out cross-validation
- **LPM** — linear probability model
- **LR** — likelihood ratio
- **LS** — least squares
- **LSDV** — least-squares dummy variable
- **MA** — moving average
- **MAR** — missing at random

- **MCAR** — missing completely at random
- **MCMC** — Markov chain Monte Carlo
- **MD** — minimum distance
- **ME** — marginal effect
- **MEM** — marginal effect at mean
- **MER** — marginal effect at representative value
- **MG** — mean group
- **MH** — Metropolis–Hastings
- **ML** — maximum likelihood
- **MLE** — maximum likelihood estimator
- **MLT** — multilevel treatment
- **MM** — method of moments
- **MNAR** — missing not at random
- **MNL** — multinomial logit
- **MNP** — multinomial probit
- **MSE** — mean squared error
- **MSL** — maximum simulated likelihood
- **MSS** — model sum of squares
- **MTE** — marginal treatment effect
- **NB** — negative binomial
- **NB1** — negative binomial variance linear in mean
- **NB2** — negative binomial variance quadratic in mean
- **NL** — nested logit
- **NLS** — nonlinear least squares
- **NNM** — nearest-neighbor matching
- **NR** — Newton–Raphson
- **NSW** — National Supported Work
- **OHIE** — Oregon Health Insurance Experiment
- **OHP** — Oregon Health Program
- **OLS** — ordinary least squares
- **PA** — population averaged
- **PFGLS** — pooled feasible generalized least squares
- **PH** — proportional hazards
- **PM** — predictive mean
- **POM** — potential-outcome mean
- **PSID** — Panel Study of Income Dynamics
- **PSM** — propensity-score matching

- **PSU** — primary sampling unit
- **QCR** — quantile count regression
- **QR** — quantile regression
- **QTE** — quantile treatment effect
- **RA** — regression adjustment
- **RCT** — randomized control trials
- **RD** — regression discontinuity
- **RE** — random effects
- **RIF** — recentered influence function
- **RMSE** — root mean squared error
- **ROC** — receiver operator characteristics
- **RPL** — random-parameters logit
- **RSS** — residual sum of squares
- **SAR** — spatial autoregressive
- **SARAR** — autoregressive spatial autoregressive
- **SEM** — structural equation model
- **SJ** — *Stata Journal*
- **SRD** — sharp regression discontinuity
- **STB** — *Stata Technical Bulletin*
- **SUR** — seemingly unrelated regressions
- **TE** — treatment effect
- **TSS** — total sum of squares
- **VCE** — variance-covariance matrix of the estimator
- **WLS** — weighted least squares
- **ZINB** — zero-inflated negative binomial
- **ZIP** — zero-inflated Poisson
- **ZTNB** — zero-truncated negative binomial
- **ZTP** — zero-truncated Poisson

# References

- Abadie, A., S. Athey, G. W. Imbens, and J. M. Wooldridge. 2022. When should you adjust standard errors for clustering? ArXiv Working Paper No. arXiv:1710.02926. <https://doi.org/10.48550/arXiv.1710.02926>.
- Abrigo, M. R. M., and I. Love. 2016. Estimation of panel vector autoregression in Stata. *Stata Journal* 16: 778–804. <https://doi.org/10.1177/1536867X1601600314>.
- Ahamada, I., and E. Flachaire. 2010. *Non-Parametric Econometrics*. Oxford: Oxford University Press.
- Anatolyev, S., and A. Skolkova. 2019. Many instruments: Implementation in Stata. *Stata Journal* 19: 849–866. <https://doi.org/10.1177/1536867X19893627>.
- Anderson, M. L., J. A. Skinner, J. P. Felmlee, R. A. Berger, and K. K. Amrami. 2008. Diagnostic comparison of 1.5 Tesla and 3.0 Tesla preoperative MRI of the wrist in patients with ulnar-sided wrist pain. *Journal of Hand Surgery* 33: 1153–1159. <https://doi.org/10.1016/j.jhsa.2008.02.028>.
- Anderson, T. W., and C. Hsiao. 1981. Estimation of dynamic models with error components. *Journal of the American Statistical Association* 76: 598–606. <https://doi.org/10.2307/2287517>.
- Anderson, T. W., and H. Rubin. 1949. Estimation of the parameters of a single equation in a complete system of stochastic equations. *Annals of Mathematical Statistics* 20: 46–63. <https://doi.org/10.1214/aoms/117730090>.
- Andrews, D. W. K. 1988. Chi-square diagnostic tests for econometric models: Introduction and applications. *Journal of Econometrics* 37: 135–156. [https://doi.org/10.1016/0304-4076\(88\)90079-6](https://doi.org/10.1016/0304-4076(88)90079-6).
- Andrews, D. W. K., and M. Buchinsky. 2000. A three-step method for choosing the number of bootstrap replications. *Econometrica* 68: 23–51. <https://doi.org/10.1111/1468-0262.00092>.

- Andrews, D. W. K., M. J. Moreira, and J. H. Stock. 2007. Performance of conditional Wald tests in IV regression with weak instruments. *Journal of Econometrics* 139: 116–132.  
<https://doi.org/10.1016/j.jeconom.2006.06.007>.
- Andrews, I., J. H. Stock, and L. Sun. 2019. Weak instruments in instrumental variables regression: Theory and practice. *Annual Review of Economics* 11: 727–753. <https://doi.org/10.1146/annurev-economics-080218-025643>.
- Angrist, J. D., G. W. Imbens, and A. B. Krueger. 1999. Jackknife instrumental variables estimation. *Journal of Applied Econometrics* 14: 57–67. [https://doi.org/10.1002/\(SICI\)1099-1255\(199901/02\)14:1%3C57::AID-JAE501%3E3.0.CO;2-G](https://doi.org/10.1002/(SICI)1099-1255(199901/02)14:1%3C57::AID-JAE501%3E3.0.CO;2-G).
- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Aranow, P. M., C. Samii, and V. A. Assenova. 2015. Cluster-robust variance estimation for dyadic data. *Political Analysis* 23: 564–577. <https://doi.org/10.1093/pan/mpv018>.
- Arellano, M. 2003. *Panel Data Econometrics*. New York: Oxford University Press.
- Arellano, M., and S. Bond. 1991. Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *Review of Economic Studies* 58: 277–297.  
<https://doi.org/10.2307/2297968>.
- Arellano, M., and O. Bover. 1995. Another look at the instrumental variable estimation of error-components models. *Journal of Econometrics* 68: 29–51. [https://doi.org/10.1016/0304-4076\(94\)01642-D](https://doi.org/10.1016/0304-4076(94)01642-D).
- Azevedo, J. P. 2004. grqreg: Stata module to graph the coefficients of a quantile regression. Statistical Software Components S437001, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s437001.html>.

- Bai, J. 2009. Panel data models with interactive fixed effects. *Econometrica* 77: 1229–1279. <https://doi.org/10.3982/ECTA6135>.
- Baltagi, B. H. 2021. *Econometric Analysis of Panel Data*. 6th ed. Cham, Switzerland: Springer.
- Baltagi, B. H., J. M. Griffin, and W. Xiong. 2000. To pool or not to pool: Homogeneous versus heterogeneous estimators applied to cigarette demand. *Review of Economics and Statistics* 82: 117–126. <https://doi.org/10.1162/003465300558551>.
- Baltagi, B. H., and S. Khanti-Akom. 1990. On efficient estimation with panel data: An empirical comparison of instrumental variables. *Journal of Applied Econometrics* 5: 401–406. <https://doi.org/10.1002/jae.3950050408>.
- Bartus, T. 2005. Estimation of marginal effects using margeff. *Stata Journal* 5: 309–329. <https://doi.org/10.1177/1536867X0500500303>.
- Baum, C. F. 2016. *An Introduction to Stata Programming*. 2nd ed. College Station, TX: Stata Press.
- Baum, C. F., M. E. Schaffer, and S. Stillman. 2007. Enhanced routines for instrumental variables/generalized method of moments estimation and testing. *Stata Journal* 7: 465–506. <https://doi.org/10.1177/1536867X0800700402>.
- Beck, N., and J. N. Katz. 1995. What to do (and not to do) with time-series cross-section data. *American Political Science Review* 89: 634–647. <https://doi.org/10.2307/2082979>.
- Benjamin, D. J., J. O. Berger, M. Johannesson, B. A. Nosek, E.-J. Wagenmakers, R. Berk, K. A. Bollen, et al.. 2018. Redefine Statistical Significance. *Nature Human Behaviour* 2: 6–10. <https://doi.org/10.1038/s41562-017-0189-z>.
- Benjamini, Y., and Y. Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B* 57: 289–300. <https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>.

- Benjamini, Y., and D. Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics* 29: 1165–1188. <https://doi.org/10.1214/aos/1013699998>.
- Bertrand, M., E. Duflo, and S. Mullainathan. 2004. How much should we trust differences-in-differences estimates? *Quarterly Journal of Economics* 119: 249–275.  
<https://doi.org/10.1162/003355304772839588>.
- Bhattacharya, D. 2005. Asymptotic inference from multi-stage samples. *Journal of Econometrics* 126: 145–171.  
<https://doi.org/10.1016/j.jeconom.2004.01.002>.
- Bitler, M. P., J. B. Gelbach, and H. W. Hoynes. 2006. What mean impacts miss: Distributional effects of welfare reform experiments. *American Economic Review* 96: 988–1012. <https://doi.org/10.1257/aer.96.4.988>.
- Blackburne, E. F., III, and M. W. Frank. 2007. Estimation of nonstationary heterogeneous panels. *Stata Journal* 7: 197–208.  
<https://doi.org/10.1177/1536867X0700700204>.
- Blomquist, S., and M. Dahlberg. 1999. Small sample properties of LIML and jackknife IV estimators: Experiments with weak instruments. *Journal of Applied Econometrics* 14: 69–88.  
[https://doi.org/10.1002/\(SICI\)1099-1255\(199901/02\)14:1%3C69::AID-JAE521%3E3.0.CO;2-7](https://doi.org/10.1002/(SICI)1099-1255(199901/02)14:1%3C69::AID-JAE521%3E3.0.CO;2-7).
- Blundell, R., and S. Bond. 1998. Initial conditions and moment restrictions in dynamic panel data models. *Journal of Econometrics* 87: 115–143. [https://doi.org/10.1016/S0304-4076\(98\)00009-8](https://doi.org/10.1016/S0304-4076(98)00009-8).
- Bound, J., D. A. Jaeger, and R. M. Baker. 1995. Problems with instrumental variables estimation when the correlation between the instruments and the endogenous explanatory variable is weak. *Journal of the American Statistical Association* 90: 443–450.  
<https://doi.org/10.2307/2291055>.
- Brodeur, A., N. Cook, and A. Heyes. 2020. Methods matter: p-hacking and publication bias in causal analysis in economics. *American Economic Review* 110: 3634–3660.  
<https://doi.org/10.1257/aer.20190687>.

- Buntin, M. B., and A. M. Zaslavsky. 2004. Too much ado about two-part models and transformation? Comparing methods of modeling Medicare expenditures. *Journal of Health Economics* 23: 525–542. <https://doi.org/10.1016/j.jhealeco.2003.10.005>.
- Cai, Y., I. A. Canay, D. Kim, and A. M. Shaikh. Forthcoming. On the implementation of approximate randomization tests in linear models with a small number of clusters. *Journal of Econometric Methods*. <https://doi.org/10.1515/jem-2021-0030>.
- Cameron, A. C., J. B. Gelbach, and D. L. Miller. 2008. Bootstrap-based improvements for inference with clustered errors. *Review of Economics and Statistics* 90: 414–427. <https://doi.org/10.1162/rest.90.3.414>.
- \_\_\_\_\_. 2011. Robust inference with multiway clustering. *Journal of Business and Economic Statistics* 29: 238–249. <https://doi.org/10.1198/jbes.2010.07136>.
- Cameron, A. C., and D. L. Miller. 2014. Robust inference for dyadic data. [http://cameron.econ.ucdavis.edu/research/dyadic Cameron\\_miller\\_december2014\\_with\\_tables.pdf](http://cameron.econ.ucdavis.edu/research/dyadic Cameron_miller_december2014_with_tables.pdf).
- \_\_\_\_\_. 2015. A practitioner’s guide to cluster–robust inference. *Journal of Human Resources* 50: 317–372. <https://doi.org/10.3388/jhr.50.2.317>.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeometrics: Methods and Applications*. Cambridge: Cambridge University Press.
- \_\_\_\_\_. 2013. *Regression Analysis of Count Data*. 2nd ed. Cambridge: Cambridge University Press.
- Cameron, A. C., and F. A. G. Windmeijer. 1997. An  $R$ -squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics* 77: 329–342. [https://doi.org/10.1016/S0304-4076\(96\)01818-0](https://doi.org/10.1016/S0304-4076(96)01818-0).
- Canay, I. A., J. P. Romano, and A. M. Shaikh. 2017. Randomization tests under an approximate symmetry assumption. *Econometrica* 85: 1013–

1030. <https://doi.org/10.3982/ECTA13081>.

Canay, I. A., A. Santos, and A. M. Shaikh. 2021. The wild bootstrap with a “small” number of “large” clusters. *Review of Economics and Statistics* 103: 346–363. [https://doi.org/10.1162/rest\\_a\\_00887](https://doi.org/10.1162/rest_a_00887).

Carter, A. V., K. T. Schnepel, and D. G. Steigerwald. 2017. Asymptotic behavior of a t-test robust to cluster heterogeneity. *Review of Economics and Statistics* 99: 698–709.  
[https://doi.org/10.1162/REST\\_a\\_00639](https://doi.org/10.1162/REST_a_00639).

Cattaneo, M. D., M. Jansson, and X. Ma. 2019. Two-step estimation and inference with possibly many included covariates. *Review of Economic Studies* 86: 1095–1122. <https://doi.org/10.1093/restud/rdy053>.

Chernozhukov, V., I. Fernández-Val, S. Han, and A. Kowalski. 2019. Censored quantile instrumental-variable estimation with Stata. *Stata Journal* 19: 768–781. <https://doi.org/10.1177/1536867X19893615>.

Chernozhukov, V., and C. B. Hansen. 2008. Instrumental variable quantile regression: A robust inference approach. *Journal of Econometrics* 142: 379–398.  
<https://doi.org/10.1016/j.jeconom.2007.06.005>.

Chiang, H., B. E. Hansen, and Y. Sasaki. 2021. Standard errors for two-way clustering with serially correlated time effects. ArXiv Working Paper No. arXiv:2201.11304.  
<https://doi.org/10.48550/arXiv.2201.11304>.

Choi, J., J. Gu, and S. Shen. 2018. Weak-instrument robust inference for two-sample instrumental variables regression. *Journal of Applied Econometrics* 33: 109–125. <https://doi.org/10.1002/jae.2580>.

Cornelissen, T. 2008. The Stata command felsdvreg to fit a linear model with two high-dimensional fixed effects. *Stata Journal* 8: 170–189.  
<https://doi.org/10.1177/1536867X0800800202>.

Cornwell, C., and P. Rupert. 1988. Efficient estimation with panel data: An empirical comparison of instrumental variables estimators. *Journal of Applied Econometrics* 3: 149–155.  
<https://doi.org/10.1002/jae.3950030206>.

- Cox, N. J. 2005. Speaking Stata: The protean quantile plot. *Stata Journal* 5: 442–460. <https://doi.org/10.1177/1536867X0500500312>.
- Cragg, J. G., and S. G. Donald. 1993. Testing identifiability and specification in instrumental variable models. *Econometric Theory* 9: 222–240. <https://doi.org/10.1017/S0266466600007519>.
- Cribari-Neto, F., and S. G. Zarkos. 1999. Bootstrap methods for heteroskedastic regression models: Evidence on estimation and testing. *Econometric Reviews* 18: 211–228.  
<https://doi.org/10.1080/07474939908800440>.
- Cruz-Gonzalez, M., I. Fernández-Val, and M. Weidner. 2017. Bias corrections for probit and logit models with two-way fixed effects. *Stata Journal* 17: 517–545.  
<https://doi.org/10.1177/1536867X1701700301>.
- Davidson, J. 2000. *Econometric Theory*. Oxford: Blackwell.
- Davidson, R., and J. G. MacKinnon. 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- \_\_\_\_\_. 2006. The case against JIVE. *Journal of Applied Econometrics* 21: 827–833. <https://doi.org/10.1002/jae.873>.
- \_\_\_\_\_. 2010. Wild bootstrap tests for IV regression. *Journal of Business and Economic Statistics* 28: 128–144.  
<https://doi.org/10.1198/jbes.2009.07221>.
- Davison, A. C., and D. V. Hinkley. 1997. *Bootstrap Methods and Their Application*. Cambridge: Cambridge University Press.
- Deb, P. 2007. fmm: Stata module to estimate finite mixture models. Statistical Software Components S456895, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s456895.html>.
- Deb, P., M. K. Munkin, and P. K. Trivedi. 2006. Private insurance, selection, and health care use: A Bayesian analysis of a Roy-type model. *Journal of Business and Economic Statistics* 24: 403–415.  
<https://doi.org/10.1198/073500106000000323>.

- Dehejia, R. H., and S. Wahba. 1999. Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. *Journal of the American Statistical Association* 94: 1053–1062.  
<https://doi.org/10.2307/2669919>.
- \_\_\_\_\_. 2002. Propensity score-matching methods for nonexperimental causal studies. *Review of Economics and Statistics* 84: 151–161. <https://doi.org/10.1162/003465302317331982>.
- Donald, S. G., and K. Lang. 2007. Inference with difference-in-differences and other panel data. *Review of Economics and Statistics* 89: 221–233. <https://doi.org/10.1162/rest.89.2.221>.
- Driscoll, J. C., and A. C. Kraay. 1998. Consistent covariance matrix estimation with spatially dependent panel data. *Review of Economics and Statistics* 80: 549–560. <https://doi.org/10.1162/003465398557825>.
- Drukker, D. M. 2014. Using gmm to solve two-step estimation problems. The Stata Blog: Not Elsewhere Classified.  
<https://blog.stata.com/2014/12/08/using-gmm-to-solve-two-step-estimation-problems/>.
- Duan, N. 1983. Smearing estimate: A nonparametric retransformation method. *Journal of the American Statistical Association* 78: 605–610.  
<https://doi.org/10.2307/2288126>.
- Eberhardt, M. 2012. Estimating panel time-series models with heterogeneous slopes. *Stata Journal* 12: 61–71.  
<https://doi.org/10.1177/1536867X1201200105>.
- Eberhardt, M., and F. Teal. 2010. Productivity analysis in global manufacturing production. Discussion Paper 515, Department of Economics, University of Oxford.  
<http://www.economics.ox.ac.uk/research/WP/pdf/paper515.pdf>.
- Efron, B., and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. New York: Chapman & Hall/CRC.
- Fafchamps, M., and F. Gubert. 2007. The formation of risk sharing networks. *Journal of Development Economics* 83: 326–350.  
<https://doi.org/10.1016/j.jdeveco.2006.05.005>.

- Farcomeni, A. 2008. A review of modern multiple hypothesis testing, with particular attention to the false discovery proportion. *Statistical Methods in Medical Research* 17: 347–388.  
<https://doi.org/10.1177/0962280206079046>.
- Fernández-Val, I., and M. Weidner. 2018. Fixed effects estimation of large-T panel data models. *Annual Review of Economics* 10: 109–138.  
<https://doi.org/10.1146/annurev-economics-080217-053542>.
- Finlay, K., and L. M. Magnusson. 2009. Implementing weak-instrument robust tests for a general class of instrumental-variables models. *Stata Journal* 9: 398–421. <https://doi.org/10.1177/1536867X0900900304>.
- Finlay, K., L. M. Magnusson, and M. E. Schaffer. 2014. weakiv10: Stata module to perform weak-instrument-robust tests and confidence intervals for instrumental-variable (IV) estimation of linear, probit and tobit models. Statistical Software Components S457910, Department of Economics, Boston College.  
<https://econpapers.repec.org/software/bocbocode/s457910.htm>.
- Fortin, N. M., T. Lemieux, and S. Firpo. 2011. Decomposition methods in economics. In Vol. 4A of *Handbook of Labor Economics*, ed. O. Ashenfelter and D. Card, 1–102. Amsterdam: Elsevier.  
[https://doi.org/10.1016/S0169-7218\(11\)00407-2](https://doi.org/10.1016/S0169-7218(11)00407-2).
- Frühwirth-Schnatter, S. 2006. *Finite Mixture and Markov Switching Models*. New York: Springer.
- Fuller, W. A. 1977. Some properties of a modification of the limited information estimator. *Econometrica* 46: 939–953.  
<https://doi.org/10.2307/1912683>.
- Gallup, J. L. 2012a. A new system for formatting estimation tables. *Stata Journal* 12: 3–28. <https://doi.org/10.1177/1536867X1201200102>.
- \_\_\_\_\_. 2012b. A programmer’s command to build formatted statistical tables. *Stata Journal* 12: 655–673.  
<https://doi.org/10.1177/1536867X1201200406>.
- Godfrey, L. G., and M. R. Wickens. 1981. Testing linear and log-linear regressions for functional form. *Review of Economic Studies* 48: 487–

496. <https://doi.org/10.2307/2297160>.

Goldsmith-Pinkham, P., I. Sorkin, and H. Swift. 2020. Bartik instruments: What, when, why and how. *American Economic Review* 110: 2586–2624. <https://doi.org/10.1257/aer.20181047>.

Goldstein, H. 1987. Multilevel covariance component models. *Biometrika* 74: 430–431. <https://doi.org/10.1093/biomet/74.2.430>.

Gomez, M. 2017. regife: Stata module to estimate linear models with interactive fixed effects. Statistical Software Component S458042, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s458042.html>.

Gould, W. W. 2018. *The Mata Book: A Book for Serious Programmers and Those Who Want to Be*. College Station, TX: Stata Press.

Gould, W. W., J. Pitblado, and B. P. Poi. 2010. *Maximum Likelihood Estimation with Stata*. 4th ed. College Station, TX: Stata Press.

Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice Hall.

\_\_\_\_\_. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Gu, A., and H. I. Yoo. 2019. vcemway: A one-stop solution for robust inference with multiway clusters. *Stata Journal* 19: 900–912.  
<https://doi.org/10.1177/1536867X19893637>.

Guimarães, P., and P. Portugal. 2010. A simple feasible procedure to fit models with high-dimensional fixed effects. *Stata Journal* 10: 628–649. <https://doi.org/10.1177/1536867X1101000406>.

Hagemann, A. 2020. Inference with a single treated cluster. ArXiv Working Paper No. arXiv:2010.04076.  
<https://doi.org/10.48550/arXiv.2010.04076>.

Hahn, J., and J. Hausman. 2002. A new specification test for the validity of instrumental variables. *Econometrica* 70: 163–189.  
<https://doi.org/10.1111/1468-0262.00272>.

Hall, A. 1987. The information matrix test for the linear model. *Review of Economic Studies* 54: 257–263. <https://doi.org/10.2307/2297515>.

- Hansen, B. E. 2022a. *Econometrics*. Princeton: Princeton University Press.
- \_\_\_\_\_. 2022b. *Probability and Statistics for Economists*. Princeton: Princeton University Press.
- Hardin, J. W., and J. M. Hilbe. 2018. *Generalized Linear Models and Extensions*. 4th ed. College Station, TX: Stata Press.
- Hastie, T. J., R. J. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.
- Heckman, J. J., and E. J. Vytlacil. 2007. Econometric evaluation of social programs, part II: Using the marginal treatment effect to organize alternative econometric estimators to evaluate social programs, and to forecast their effects in new environments. In Vol. 6B of *Handbook of Econometrics*, ed. J. J. Heckman and E. E. Leamer, 4875–5143. Amsterdam: Elsevier. [https://doi.org/10.1016/S1573-4412\(07\)06071-0](https://doi.org/10.1016/S1573-4412(07)06071-0).
- Hersch, J. 1998. Compensating differentials for gender-specific job injury risks. *American Economic Review* 88: 598–607.
- Hoechle, D. 2007. Robust standard errors for panel regressions with cross-sectional dependence. *Stata Journal* 7: 281–312. <https://doi.org/10.1177/1536867X0700700301>.
- Holtz-Eakin, D., W. K. Newey, and H. S. Rosen. 1988. Estimating vector autoregressions with panel data. *Econometrica* 56: 1371–1395. <https://doi.org/10.2307/1913103>.
- Horowitz, J. L. 2001. The bootstrap. In Vol. 5 of *Handbook of Econometrics*, ed. J. J. Heckman and E. Leamer, 3159–3228. Amsterdam: Elsevier. [https://doi.org/10.1016/S1573-4412\(01\)05005-X](https://doi.org/10.1016/S1573-4412(01)05005-X).
- Hosmer, D. W., Jr., S. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.
- Hsiao, C. 2014. *Analysis of Panel Data*. 3rd ed. Cambridge: Cambridge University Press.

- Huber, P. J. 1967. The behavior of maximum likelihood estimates under nonstandard conditions. In Vol. 1 of *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 221–233. Berkeley, CA: University of California Press.
- Huber, P. J., and E. M. Ronchetti. 2009. *Robust Statistics*. 2nd ed. New York: Wiley.
- Huettner, F., and M. Sunder. 2012. rego: Stata module for decomposing goodness of fit according to Owen and Shapley values. UK Stata Users Group meeting proceedings.  
[https://www.stata.com/meeting/uk12/abstracts/materials/uk12\\_sunder.pdf](https://www.stata.com/meeting/uk12/abstracts/materials/uk12_sunder.pdf)
- Ibragimov, R., and U. K. Müller. 2010. *t*-statistic based correlation and heterogeneity robust inference. *Journal of Business and Economic Statistics* 28: 453–468. <https://doi.org/10.1198/jbes.2009.08046>.
- Imbens, G. W., and D. B. Rubin. 2015. *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge: Cambridge University Press.
- Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation* 3: 79–87. <https://doi.org/10.1162/neco.1991.3.1.79>.
- James, G., D. Witten, T. J. Hastie, and R. J. Tibshirani. 2021. *An Introduction to Statistical Learning: With Applications in R*. 2nd ed. New York: Springer.
- Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308. <https://doi.org/10.1177/1536867X0500500302>.
- \_\_\_\_\_. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244. <https://doi.org/10.1177/1536867X0700700207>.
- \_\_\_\_\_. 2008. The Blinder–Oaxaca decomposition for linear regression models. *Stata Journal* 8: 453–479. <https://doi.org/10.1177/1536867X0800800401>.
- Jordan, M. I., and R. A. Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6: 181–214.

<https://doi.org/10.1162/neco.1994.6.2.181>.

Karabiyik, H., F. C. Palm, and J.-P. Urbain. 2019. Econometric analysis of panel data models with multifactor error structures. *Annual Review of Economics* 11: 495–522. <https://doi.org/10.1146/annurev-economics-063016-104338>.

Kleibergen, F. 2005. Testing parameters in GMM without assuming that they are identified. *Econometrica* 73: 1103–1123.  
<https://doi.org/10.1111/j.1468-0262.2005.00610.x>.

Kleibergen, F., and R. Paap. 2006. Generalized reduced rank tests using the singular value decomposition. *Journal of Econometrics* 133: 97–126. <https://doi.org/10.1016/j.jeconom.2005.02.011>.

Kline, P., and A. Santos. 2012. A score based approach to wild bootstrap inference. *Journal of Econometric Methods* 1: 23–41.  
<https://doi.org/10.1515/2156-6674.1006>.

Koenker, R. 2005. *Quantile Regression*. New York: Cambridge University Press.

Kozbur, D. 2020. Analysis of testing-based forward model selection. *Econometrica* 88: 2147–2173. <https://doi.org/10.3982/ECTA16273>.

Kreuter, F., and R. Valliant. 2007. A survey on survey statistics: What is done and can be done in Stata. *Stata Journal* 7: 1–21.  
<https://doi.org/10.1177/1536867X0700700101>.

LaLonde, R. J. 1986. Evaluating the econometric evaluations of training programs with experimental data. *American Economic Review* 76: 604–620.

Lane, P. W., and J. A. Nelder. 1982. Analysis of covariance and standardization as instances of prediction. *Biometrics* 38: 613–621.  
<https://doi.org/10.2307/2530043>.

Lee, C. H., and D. G. Steigerwald. 2018. Inference for clustered data. *Stata Journal* 18: 447–460.  
<https://doi.org/10.1177/1536867X1801800210>.

- Lee, D. S., J. McCrary, M. J. Moreira, and J. R. Porter. 2021. Valid t-ratio inference for IV. NBER Working Paper No. 29124, The National Bureau of Economic Research. <https://doi.org/10.3386/w29124>.
- Lee, M. 2002. *Panel Data Econometrics: Methods-of-Moments and Limited Dependent Variables*. San Diego: Academic Press.
- Levin, A., C.-F. Lin, and C.-S. J. Chu. 2002. Unit root tests in panel data: Asymptotic and finite-sample properties. *Journal of Econometrics* 108: 1–24. [https://doi.org/10.1016/S0304-4076\(01\)00098-7](https://doi.org/10.1016/S0304-4076(01)00098-7).
- Li, Q., and J. S. Racine. 2007. *Nonparametric Econometrics: Theory and Practice*. Princeton, NJ: Princeton University Press.
- Liang, K.-Y., and S. L. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* 73: 13–22. <https://doi.org/10.1093/biomet/73.1.13>.
- List, J. A., A. M. Shaikh, and Y. Xu. 2019. Multiple hypothesis testing in experimental economics. *Experimental Economics* 22: 773–793. <https://doi.org/10.1007/s10683-018-09597-5>.
- Long, J. S., and J. Freese. 2014. *Regression Models for Categorical Dependent Variables Using Stata*. 3rd ed. College Station, TX: Stata Press.
- Machado, J. A. F., P. M. D. C. Parente, and J. M. C. Santos Silva. 2011. qreg2: Stata module to perform quantile regression with robust and clustered standard errors. Statistical Software Components S457369, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s457369.html>.
- MacKinnon, J. G. 2002. Bootstrap inference in econometrics. *Canadian Journal of Economics* 35: 615–645. <https://doi.org/10.1111/0008-4085.00147>.
- MacKinnon, J. G., M. Ø. Nielsen, and M. D. Webb. 2022. Leverage, influence, and the jackknife in clustered regression models: Reliable inference using summclust. Queen’s University, Department of Economics, Working Paper No. 1483. <https://ideas.repec.org/p/qed/wpaper/1483.html>.

- \_\_\_\_\_. Forthcoming. Cluster-robust inference: A guide to empirical practice. *Journal of Econometrics*.  
<https://doi.org/10.1016/j.jeconom.2022.04.001>.
- MacKinnon, J. G., and M. D. Webb. 2020. When and how to deal with clustered errors in regression models. Technical report, Queen's University, Department of Economics, Working Paper No. 1421.  
<https://ideas.repec.org/p/qed/wpaper/1421.html>.
- Magnusson, L. M. 2010. Inference in limited dependent variable models robust to weak identification. *Econometrics Journal* 13: S56–S79.  
<https://doi.org/10.1111/j.1368-423X.2009.00309.x>.
- Mander, A., and D. Clayton. 1999. sg116: Hotdeck imputation. *Stata Technical Bulletin* 51: 32–34. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 9, pp. 196–199. College Station, TX: Stata Press.
- Manjón, M., and O. Martínez. 2014. The chi-squared goodness-of-fit test for count-data models. *Stata Journal* 14: 798–816.  
<https://doi.org/10.1177/1536867X1401400406>.
- Manski, C. F. 1993. Identification of endogenous social effects: The reflection problem. *Review of Economic Studies* 60: 531–542.  
<https://doi.org/10.2307/2298123>.
- McCaffrey, D. F., J. R. Lockwood, K. Mihaly, and T. R. Sass. 2012. A review of Stata commands for fixed-effects estimation in normal linear models. *Stata Journal* 12: 406–432.  
<https://doi.org/10.1177/1536867X1201200305>.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.
- Miglioretti, D. L., and P. J. Heagarty. 2006. Marginal modeling of nonnested multilevel data using standard software. *American Journal of Epidemiology* 165: 453–463. <https://doi.org/10.1093/aje/kwk020>.
- Mikusheva, A. 2010. Robust confidence sets in the presence of weak instruments. *Journal of Econometrics* 157: 236–247.  
<https://doi.org/10.1016/j.jeconom.2009.12.003>.

- Mikusheva, A., and B. P. Poi. 2006. Tests and confidence sets with correct size when instruments are potentially weak. *Stata Journal* 6: 335–347. <https://doi.org/10.1177/1536867X0600600303>.
- Miller, G. E. 1991. Asymptotic test statistics for coefficients of variation. *Communications in Statistics—Theory and Methods* 20: 3351–3363. <https://doi.org/10.1080/03610929108830707>.
- Mitchell, M. N. 2022. *A Visual Guide to Stata Graphics*. 4th ed. College Station, TX: Stata Press.
- Montiel Olea, J. L., and C. E. Pflueger. 2013. A robust test for weak instruments. *Journal of Business and Economic Statistics* 31: 358–369. <https://doi.org/10.1080/00401706.2013.806694>.
- Moreira, M. J. 2003. A conditional likelihood ratio test for structural models. *Econometrica* 71: 1027–1048. <https://doi.org/10.1111/1468-0262.00438>.
- Mundlak, Y. 1978. On the pooling of time series and cross section data. *Econometrica* 46: 69–85. <https://doi.org/10.2307/1913646>.
- Nagar, A. L. 1959. The bias and moment matrix of the general k-class estimators of the parameters in simultaneous equations. *Econometrica* 27: 575–595. <https://doi.org/10.2307/1909352>.
- Newey, W. K., and K. D. West. 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708. <https://doi.org/10.2307/1913610>.
- Newson, R. B. 2012. Sensible parameters for univariate and multivariate splines. *Stata Journal* 12: 479–504. <https://doi.org/10.1177/1536867X1201200310>.
- Newson, R. B., and the ALSPAC Study Team. 2003. Multiple-test procedures and smile plots. *Stata Journal* 3: 109–132. <https://doi.org/10.1177/1536867X0300300202>.
- Pacini, D., and F. A. G. Windmeijer. 2016. Robust inference for the two-sample 2SLS estimator. *Economics Letters* 146: 50–54. <https://doi.org/10.1016/j.econlet.2016.06.033>.

- Pagan, A., and A. Ullah. 1999. *Nonparametric Econometrics*. Cambridge: Cambridge University Press.
- Papps, K. L. 2006. outsum: Stata module to write formatted descriptive statistics to a text file. Statistical Software Components S456780, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s456780.html>.
- Parente, P. M. D. C., and J. M. C. Santos Silva. 2016. Quantile regression with clustered data. *Journal of Econometric Methods* 5: 1–15.  
<https://doi.org/10.1515/jem-2014-0011>.
- Pesaran, M. H. 2006. Estimation and inference in large heterogeneous panels with a multifactor error structure. *Econometrica* 74: 967–1012.  
<https://doi.org/10.1111/j.1468-0262.2006.00692.x>.
- \_\_\_\_\_. 2015. *Time Series and Panel Data Econometrics*. Oxford: Oxford University Press.
- Pesaran, M. H., and R. Smith. 1995. Estimating long-run relationships from dynamic heterogeneous panels. *Journal of Econometrics* 68: 79–113. [https://doi.org/10.1016/0304-4076\(94\)01644-F](https://doi.org/10.1016/0304-4076(94)01644-F).
- Pflueger, C. E., and S. Wang. 2015. weakivtest: Stata module to perform weak instrument test for a single endogenous regressor in TSLS and LIML. Statical Software Component S457732, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s457732.html>.
- Poi, B. P. 2004. From the help desk: Some bootstrapping techniques. *Stata Journal* 4: 312–328.  
<https://doi.org/10.1177/1536867X0400400308>.
- \_\_\_\_\_. 2006. Jackknife instrumental variables estimation in Stata. *Stata Journal* 6: 364–376.  
<https://doi.org/10.1177/1536867X0600600305>.
- Politis, D. N., J. P. Romano, and M. Wolf. 1999. *Subsampling*. New York: Springer.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes in C: The Art of Scientific Computing*. 3rd ed.

Cambridge: Cambridge University Press.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Romano, J. P., and M. Wolf. 2005. Stepwise multiple testing as formalized data snooping. *Econometrica* 73: 1237–1282.  
<https://doi.org/10.1111/j.1468-0262.2005.00615.x>.

Roodman, D. 2009. How to do xtabond2: An introduction to difference and system GMM in Stata. *Stata Journal* 9: 86–136.  
<https://doi.org/10.1177/1536867X0900900106>.

Roodman, D., M. Ø. Nielsen, J. G. MacKinnon, and M. D. Webb. 2019. Fast and wild: Bootstrap inference in Stata using boottest. *Stata Journal* 19: 4–60. <https://doi.org/10.1177/1536867X19830877>.

Royston, P., and G. Ambler. 1998. sg79: Generalized additive models. *Stata Technical Bulletin* 42: 38–43. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 7, pp. 217–224. College Station, TX: Stata Press.

Salgado-Ugarte, I., M. Shimizu, and T. Taniuchi. 1996. snp10: Nonparametric regression: Kernel, WARP, and k-NN estimators. *Stata Technical Bulletin* 30: 15–30. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 5, pp. 197–218. College Station, TX: Stata Press.

Sanderson, E., and F. A. G. Windmeijer. 2016. A weak instrument  $F$ -test in linear IV models with multiple endogenous variables. *Journal of Econometrics* 190: 212–221.  
<https://doi.org/10.1016/j.jeconom.2015.06.004>.

Schaffer, M. E. 2005. xtivreg2: Stata module to perform extended IV/2SLS, GMM and AC/HAC, LIML, and  $k$ -class regression for panel-data models. Statistical Software Components S456501, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s456501.html>.

Schaffer, M. E., and S. Stillman. 2006. xtoverid: Stata module to calculate tests of overidentifying restrictions after xtreg, xtivreg, xtivreg2, and xtaylor. Statistical Software Components S456779, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s456779.html>.

- Shi, X. 2015. A nondegenerate Vuong test. *Quantitative Economics* 6: 85–121. <https://doi.org/10.3982/QE382>.
- Shorrocks, A. F. 2013. Decomposition procedures for distributional analysis: A unified framework based on the Shapley value. *Journal of Economic Inequality* 11: 99–126. <https://doi.org/10.1007/s10888-011-9214-z>.
- Solon, G., S. J. Haider, and J. M. Wooldridge. 2015. What are we weighting for? *Journal of Human Resources* 50: 301–316. <https://doi.org/10.3388/jhr.50.2.301>.
- Staiger, D., and J. H. Stock. 1997. Instrumental variables regression with weak instruments. *Econometrica* 65: 557–586. <https://doi.org/10.2307/2171753>.
- Stock, J. H., and J. H. Wright. 2000. GMM with weak identification. *Econometrica* 68: 1055–1096. <https://doi.org/10.1111/1468-0262.00151>.
- Stock, J. H., and M. Yogo. 2005. Testing for weak instruments in linear IV regression. In *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, ed. D. W. K. Andrews and J. H. Stock, 80–108. New York: Cambridge University Press.
- Tabord-Meehan, M. 2019. Inference with dyadic data: Asymptotic behavior of the dyadic-robust  $t$ -statistic. *Journal of Business and Economic Statistics* 37: 671–680. <https://doi.org/10.1080/07350015.2017.1409630>.
- Thompson, S. B. 2011. Simple formulas for standard errors that cluster by both firm and time. *Journal of Financial Economics* 99: 1–10. <https://doi.org/10.1016/j.jfineco.2010.08.016>.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57: 307–333. <https://doi.org/10.2307/1912557>.
- Wada, R. 2005. outreg2: Stata module to arrange regression outputs into an illustrative table. Statistical Software Components S456416,

Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s456416.html>.

- Wasserstein, R. L., and N. A. Lazar. 2016. The ASA statement on  $p$ -values: Context, process, and purpose. *American Statistician* 70: 129–133. <https://doi.org/10.1080/00031305.2016.1154108>.
- Wasserstein, R. L., A. L. Schirm, and N. A. Lazar. 2019. Statistical inference in the 21st century: A world beyond  $p < 0.05$ . *American Statistician* 73 (Suppl. 1).
- White, H. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica* 48: 817–838. <https://doi.org/10.2307/1912934>.
- \_\_\_\_\_. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50: 1–25. <https://doi.org/10.2307/1912526>.
- Windmeijer, F. A. G. 2005. A finite sample correction for the variance of linear efficient two-step GMM estimators. *Journal of Econometrics* 126: 25–51. <https://doi.org/10.1016/j.jeconom.2004.02.005>.
- Wolfe, F. 2002. fsum: Stata module to generate and format summary statistics. Statistical Software Components S426501, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s426501.html>.
- Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.
- \_\_\_\_\_. 2021. Two-way fixed effects, the two-way Mundlak regression, and difference-in-differences estimators. Working paper. [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=3906345](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3906345).
- Young, A. 2019. Channeling Fisher: Randomization tests and the statistical significance of seemingly significant experimental results. *Quarterly Journal of Economics* 134: 557–598.  
<https://doi.org/10.1093/qje/qjy029>.
- \_\_\_\_\_. Forthcoming. Consistency without inference: Instrumental variables in practical application. *European Economic Review*.  
<https://doi.org/10.1016/j.eurocorev.2022.104112>.

# Author index

## A

- Abrigo, M. R., [9.4.10](#)  
Ahamada, I., [14.1](#)  
Ambler, G., [14.5.3](#)  
Anatolyev, S., [7.9.1](#), [7.11](#)  
Anderson, M., [11.6.5](#)  
Anderson, T. W., [7.7.2](#), [9.4.2](#)  
Andrews, D. W. K., [7.7.4](#), [11.9.3](#), [12.3.4](#)  
Andrews, I., [7.5](#), [7.6.4](#), [7.7.1](#)  
Angrist, J. D., [7.5.7](#), [7.9.2](#)  
Aranow, P. M., [6.4.5](#)  
Arellano, M., [9.4.2](#), [9.4.7](#), [9.4.10](#), [9.6](#)  
Assenova, V., [6.4.5](#)  
Azevedo, J. P., [15.3.10](#), [15.6](#)

## B

- Bai, J., [9.5.6](#), [9.5.8](#)  
Baker, R. M., [7.5.7](#)  
Baltagi, B. H., [8.3.1](#), [8.11](#), [9.3.3](#), [9.5.1](#), [9.5.9](#), [9.6](#), [11.9.6](#)  
Bartus, T., [4.5.2](#), [10.4.2](#), [13.7.2](#)  
Baum, C. F., [6.4.4](#), [7.6.9](#), [7.9](#), [7.9.3](#), [7.11](#), [A.4](#), [B.4](#), [C.3](#)  
Beck, N., [9.5.4](#)  
Benjamini, Y., [11.6.4](#)  
Bertrand, M., [8.2.3](#)  
Bhattacharya, D., [6.10](#)  
Bitler, M. P., [15.5](#)  
Blackburne, E. F., [9.5.8](#)  
Blomquist, S., [7.9.2](#)  
Blundell, R., [9.4.7](#)  
Bond, S., [9.4.2](#), [9.4.7](#), [9.4.10](#)  
Bound, J. D., [7.5.7](#)  
Bover, O., [9.4.7](#)  
Buchinsky, M. Y., [12.3.4](#)

Buntin, M. B., [3.4.8](#)

## C

Cameron, A. C., [3.7](#) , [3.8.3](#) , [5.4](#) , [5.6.4](#) , [5.7](#) , [6.4.1](#) , [6.4.4](#) , [6.4.5](#) , [6.4.6](#) ,  
[6.10](#) , [7.5.7](#) , [7.7.5](#) , [8.2.3](#) , [9.6](#) , [11.9.3](#) , [11.9.6](#) , [12.5.1](#) , [12.6](#) , [12.8](#) , [12.10](#) ,  
[13.3.8](#) , [13.8.2](#) , [14.2.6](#) , [15.2.3](#)

Canay, I. A., [11.10](#)

Carter, A. V., [6.4.6](#) , [12.6.3](#)

Cattaneo, M. D., [12.9.1](#)

Chernozhukov, V., [7.7.2](#) , [15.3.11](#)

Choi, J., [7.9.4](#)

Chu, C.-S. J., [9.5.9](#)

Cornelissen, T., [6.6.6](#)

Cornwell, C., [8.3.1](#) , [9.3.3](#)

Cox, N. J., [15.3.1](#) , [15.6](#)

Cragg, J. G., [7.6.3](#)

Cribari-Neto, F., [11.5.3](#)

Cruz-Gonzalez, M., [13.9.4](#)

## D

Dahlberg, M., [7.9.2](#)

Davidson, J., [7.4.6](#)

Davidson, R., [7.5.2](#) , [7.9.2](#) , [12.6](#) , [12.6.5](#) , [12.10](#)

Davison, A. C., [12.3.6](#) , [12.6](#) , [12.10](#)

Deb, P., [10.2.1](#) , [14.2.5](#)

Dehejia, R. H., [4.8.1](#)

Donald, S. G., [6.4.6](#) , [7.6.3](#)

Driscoll, J. C., [9.5.3](#)

Drukker, D. M., [13.3.11](#)

Duan, N., [4.2.3](#)

Duflo, E., [8.2.3](#)

## E

Eberhardt, M., [9.5.8](#)

Efron, B., [12.3.4](#) , [12.3.6](#) , [12.3.9](#) , [12.5.1](#) , [12.5.3](#) , [12.10](#)

## F

Farcomeni, A., [11.6.5](#)

Fernandez-Val, I., [13.9.4](#), [15.3.11](#)  
Finlay, K., [7.7.5](#), [7.11](#)  
Firpo, S., [4.6.1](#)  
Flachaire, E., [14.1](#)  
Flannery, B. P., [5.7](#), [13.7.10](#)  
Fortin, N. M., [4.6.1](#)  
Frank, M. W., [9.5.8](#)  
Freese, J., [13.7.12](#)  
Fruehwirth-Schnatter, S., [14.2.2](#)

## G

Gallup, J.L., [3.5.7](#)  
Gelbach, J. B., [6.4.4](#), [12.6](#), [12.8](#), [12.10](#), [15.5](#)  
Godfrey, L. G., [3.3](#)  
Goldsmith-Pinkham, P., [9.2.4](#)  
Goldstein, H., [6.7.8](#)  
Gomez, M., [9.5.6](#)  
Gould, W., [A.2.8](#), [B.4](#)  
Greene, W. H., [3.10](#), [11.11](#), [13.8.2](#)  
Griffin, J. M., [9.5.1](#)  
Gu, A., [6.4.4](#)  
Gu, J., [7.9.4](#)  
Guimaraes, P., [6.4.4](#), [6.6.6](#)

## H

Hagemann, A., [6.4.6](#), [15.2.3](#)  
Hahn, J., [7.4.3](#)  
Haider, S. J., [3.8.3](#)  
Hall, A., [3.7.5](#)  
Han, S., [15.3.11](#)  
Hansen, C. B., [7.7.2](#)  
Hardin, J. W., [13.3.8](#)  
Hausman, J., [7.4.3](#)  
Heagarty, P. J., [6.4.4](#)  
Heckman, J. J., [13.7.16](#)  
Hersch, J., [12.6.3](#)  
Hilbe, J. M., [13.3.8](#)

Hinkley, D. V., [12.3.6](#), [12.6](#), [12.10](#)  
Hinton, G. E., [14.2.5](#)  
Hochberg, Y., [11.6.4](#)  
Hoechle, D., [9.5.3](#)  
Holtz-Eakin, D., [9.4.2](#), [9.4.10](#)  
Horowitz, J. L., [12.5.1](#), [12.6](#), [12.8](#), [12.10](#)  
Hosmer, D. W., Jr., [11.9.6](#)  
Hoynes, H. W., [15.5](#)  
Hsiao, C., [8.11](#), [9.4.2](#), [9.6](#)  
Huber, P. J., [3.6.1](#), [13.4.5](#)  
Huettner, F., [4.7.1](#)

## I

Ibragimov, R., [6.4.6](#)  
Imbens, G. W., [7.9.2](#), [11.10](#)

## J

Jacobs, R. A., [14.2.5](#)  
Jaeger, D. A., [7.5.7](#)  
Jann, B., [4.6.2](#)  
Jansson, M., [12.9.1](#)  
Jordan, M. I., [14.2.5](#)

## K

Katz, J. N., [9.5.4](#)  
Khanti-Akom, S., [8.3.1](#), [9.3.3](#)  
Kleibergen, F., [7.6.3](#), [7.7.5](#)  
Kline, P., [12.6](#), [12.6.4](#)  
Koenker, R., [15.2.1](#)  
Kowalski, A. E., [15.3.11](#)  
Kozbur, D., [11.3.8](#)  
Kraay, A. C., [9.5.3](#)  
Kreuter, F., [6.10](#)  
Krueger, A. B., [7.9.2](#)

## L

LaLonde, R. J., [4.8.1](#)  
Lane, P. W., [4.4](#)

Lang, K., [6.4.6](#)  
Lee, C. H., [6.4.6](#)  
Lee, D. S., [7.7.1](#)  
Lee, M., [9.6](#)  
Lemeshow, S., [11.9.6](#)  
Lemieux, T., [4.6.1](#)  
Levin, A., [9.5.9](#)  
Liang, K.-Y., [13.4.6](#)  
Lin, C.-F., [9.5.9](#)  
List, J. A., [11.6.5](#)  
Lockwood, J. R., [6.6.6](#)  
Long, J. S., [13.7.12](#)  
Love, I., [9.4.10](#)

## M

Ma, X., [12.9.1](#)  
Machado, J. A. F., [15.2.4](#), [15.3.6](#)  
MacKinnon, J. G., [3.4.6](#), [6.4.6](#), [6.10](#), [7.5.2](#), [7.9.2](#), [8.2.3](#), [12.6](#), [12.6.2](#),  
[12.6.5](#), [12.8](#), [12.10](#)  
Magnusson, L. M., [7.7.5](#), [7.11](#)  
Mander, A., [2.4.6](#)  
Manjon, M., [11.9.3](#)  
Manski, C. F., [6.6.5](#)  
Martinez, O., [11.9.3](#)  
McCaffrey, D. F., [6.6.6](#)  
McCravy, J., [7.7.1](#)  
McCullagh, P., [13.3.8](#)  
Miglioretti, D. L., [6.4.4](#)  
Mihaly, K., [6.6.6](#)  
Mikusheva, A., [7.7.4](#), [7.11](#)  
Miller, D. L., [6.4.4](#), [6.4.5](#), [6.4.6](#), [6.10](#), [8.2.3](#), [12.6](#), [12.8](#), [12.10](#)  
Miranda, A., [15.6](#)  
Mitchell, M. N., [2.7](#)  
Montiel Olea, J. L., [7.6.4](#), [7.6.5](#), [7.6.7](#), [7.6.11](#), [7.8](#)  
Moreira, M. J., [7.7.1](#), [7.7.4](#)  
Mullainathan, S., [8.2.3](#)  
Muller, R. K., [6.4.6](#)

Mundlak, Y., [8.7.4](#)

## N

Nelder, J. A., [4.4](#), [13.3.8](#)  
Newey, W. K., [9.4.2](#), [9.4.10](#), [13.4.7](#)  
Newson, R. B., [11.6.1](#), [11.6.2](#), [11.6.4](#)  
Nielsen, M. E., [3.4.6](#), [6.4.6](#), [8.2.3](#), [12.6.2](#)  
Nielsen, M. O., [6.4.6](#), [6.10](#)  
Nowlan, S. J., [14.2.5](#)

## P

Paap, R., [7.6.3](#)  
Pacini, D., [7.9.4](#)  
Papps, K. L., [3.2.4](#)  
Parente, P. M., [15.2.3](#), [15.2.4](#), [15.3.6](#)  
Pesaran, M. H., [9.5.8](#), [9.5.9](#), [9.6](#)  
Pfleuger, C., [7.6.4](#), [7.6.5](#)  
Pflueger, C., [7.6.7](#), [7.6.11](#), [7.8](#)  
Pischke, J.-S., [7.5.7](#)  
Pitblado, J., [A.2.8](#)  
Poi, B. P., [7.7.4](#), [7.9.2](#), [7.11](#), [12.3.4](#)  
Politis, D. N., [12.8.5](#)  
Porter, J., [7.7.1](#)  
Portugal, P., [6.4.4](#), [6.6.6](#)  
Press, W. H., [5.7](#), [13.7.10](#)

## R

Rabe-Hesketh, S., [6.7.8](#)  
Romano, J. P., [11.6.5](#), [11.10](#), [12.8.5](#)  
Roodman, D., [3.4.6](#), [6.4.6](#), [8.2.3](#), [9.4.9](#), [12.6.2](#)  
Rosen, H. S., [9.4.2](#), [9.4.10](#)  
Royston, P., [14.5.3](#)  
Rubin, D. B., [11.10](#)  
Rubin, H., [7.7.2](#)  
Rupert, P., [8.3.1](#), [9.3.3](#)

## S

Saami, C., [6.4.5](#)

Salgado-Ugarte, I. H., [14.6.2](#)  
Sanderson, E., [7.6.10](#)  
Santos Silva, J. M. C., [15.2.3](#), [15.2.4](#), [15.3.6](#)  
Santos, A., [12.6](#), [12.6.4](#)  
Sass, T. R., [6.6.6](#)  
Schaffer, M. E., [6.4.4](#), [7.6.9](#), [7.7.5](#), [7.9](#), [7.9.3](#), [7.11](#), [9.2.2](#)  
Schnepel, K. T., [6.4.6](#), [12.6.3](#)  
Shaikh, A. M., [11.6.5](#), [11.10](#)  
Shen, S., [7.9.4](#)  
Shimizu, M., [14.6.2](#)  
Shorrocks, A., [4.7](#)  
Skolkova, A., [7.9.1](#), [7.11](#)  
Skrondal, A., [6.7.8](#)  
Smith, R., [9.5.8](#)  
Solon, G., [3.8.3](#)  
Sorkin, I., [9.2.4](#)  
Sribney, W., [A.2.8](#)  
Staiger, D., [7.6.4](#)  
Steigerwald, D. G., [6.4.6](#), [12.6.3](#)  
Stillman, S., [6.4.4](#), [7.6.9](#), [7.9](#), [7.9.3](#), [7.11](#)  
Stock, J. H., [7.5](#), [7.6.3](#), [7.6.4](#), [7.7.1](#), [7.7.4](#), [7.7.5](#)  
Sun, L., [7.5](#), [7.6.4](#), [7.7.1](#)  
Sunder, M., [4.7.1](#)  
Swift, H., [9.2.4](#)

## T

Tabord-Mehan, M., [6.4.5](#)  
Taniuchi, T., [14.6.2](#)  
Teal, F., [9.5.8](#)  
Teukolsky, S. A., [5.7](#), [13.7.10](#)  
Thompson, S. B., [6.4.4](#)  
Tibshirani, R. J., [12.3.4](#), [12.3.6](#), [12.3.9](#), [12.5.1](#), [12.5.3](#), [12.10](#)  
Trivedi, P. K., [3.7](#), [3.8.3](#), [5.4](#), [5.6.4](#), [5.7](#), [6.4.1](#), [6.10](#), [7.5.7](#), [7.7.5](#), [9.6](#),  
[10.2.1](#), [11.9.3](#), [11.9.6](#), [12.5.1](#), [12.6](#), [12.10](#), [13.3.8](#), [13.8.2](#), [14.2.6](#),  
[15.2.3](#)

## V

Valliant, R., [6.10](#)  
Vetterling, W. T., [5.7](#), [13.7.10](#)  
Vuong, Q. H., [13.8.2](#)  
Vytlacil, E. J., [13.7.16](#)

## W

Wada, R., [3.5.7](#)  
Wahba, S., [4.8.1](#)  
Wang, S., [7.6.7](#)  
Webb, M. D., [3.4.6](#), [6.4.6](#), [6.10](#), [8.2.3](#), [12.6.2](#)  
Weidner, M., [13.9.4](#)  
West, K. D., [13.4.7](#)  
White, H., [3.4.5](#), [13.3.1](#), [13.4.5](#)  
Wickens, M. R., [3.3](#)  
Windmeijer, F. A. G., [7.6.10](#), [7.9.4](#), [9.4.4](#), [9.4.9](#)  
Wolf, M., [11.6.5](#), [12.8.5](#)  
Wolfe, F., [3.2.4](#)  
Wooldridge, J. M., [3.8.3](#), [7.7.5](#), [8.11](#), [11.11](#)  
Wright, J. H., [7.7.5](#)

## X

Xiong, W., [9.5.1](#)  
Xu, Y., [11.6.5](#)

## Y

Yogo, M., [7.6.3](#), [7.6.4](#)  
Yoo, H. I., [6.4.4](#)  
Young, A., [7.8](#), [11.10](#)

## Z

Zarkos, S., [11.5.3](#)  
Zaslavsky, A. M., [3.4.8](#)  
Zeger, S. L., [13.4.6](#)

# Subject index

## A

abbreviations, [1.3.5](#)  
ado-files, [1.12](#), [1.12](#), [A.2.8](#)  
adoupdate command, [1.12](#)  
AIC, *see* [Akaike information criterion](#)  
Akaike information criterion, [13.8.2](#), [14.3.8](#)  
AME, *see* marginal effects, average marginal effect  
analysis-of-variance table, [1.6.2](#)  
Anderson–Rubin test, [7.7.2](#), [7.7.3](#)  
anova command, [3.2.7](#)  
append command, [2.5.7](#)  
areg command, [6.6.4](#), [8.5.4](#)  
Arellano-Bond estimator, [9.4](#), [9.4.10](#)  
arithmetic operators, [1.3.6](#)  
augmented mean group estimator, [9.5.8](#)  
average  
    marginal effect, *see* [marginal effects](#)  
    treatment effect, [4.8](#)

## B

Bayesian information criterion, [13.8.2](#), [14.3.8](#)  
before–after comparison, [4.8.2](#)  
BIC, *see* [Bayesian information criterion](#)  
binary outcome models, [10.2](#), [10.5](#)  
    coefficient interpretation, [10.4.6](#)  
    introduction, [10.2](#), [10.5](#)  
    linear probability model, [10.4.7](#)  
    logit model, [10.2.2](#), [10.5](#), [10.5](#)  
    marginal effects, [10.4](#), [10.4.7](#)  
    probit model, [10.2.2](#), [10.3](#), [10.3.5](#)  
bootstrap methods, [3.4.7](#), [3.4.7](#), [3.5.11](#), [3.5.11](#), [12](#), [12.11](#)  
    asymptotic refinement, [12.2.3](#), [12.3.6](#), [12.5](#), [12.6.5](#)  
    asymptotically pivotal statistic, [12.5.1](#)

BCa confidence interval, [12.3.6](#), [12.5.1](#)  
bias estimation, [12.3.9](#)  
bias-corrected confidence interval, [12.3.6](#)  
bootstrap pairs, [3.4.7](#), [12.3](#), [12.3.1](#)  
bootstrap pairs resampling, [12.8.1](#)  
case bootstrap, [12.3.1](#)  
cluster pairs, [3.5.11](#), [3.5.11](#), [12.3.1](#), [12.3.5](#), [12.3.5](#), [12.4.1](#)  
cluster-specific indicators, [12.3.5](#)  
confidence intervals, [12.3.6](#), [12.3.8](#), [12.5](#), [12.6.5](#)  
for 2SLS, [7.8](#)  
for Hausman test, [12.4.6](#), [12.4.6](#)  
for quantile regression, [15.2.4](#)  
for two-step estimator, [12.4.5](#), [12.4.5](#)  
jackknife, [12.9.1](#), [12.9.2](#)  
misuse of bootstrap, [12.2.4](#)  
nonparametric bootstrap, [12.3.1](#)  
normal-based confidence interval, [12.3.6](#)  
number of bootstraps, [12.3.4](#)  
parametric bootstrap resampling, [12.8.2](#)  
percentile-*t* confidence interval, [12.3.6](#), [12.5.3](#)  
percentile-*t* method, [12.5.1](#), [12.6.5](#)  
residual bootstrap resampling, [12.8.3](#)  
score-based wild cluster bootstrap, [12.6.4](#), [12.6.4](#)  
standard error estimation, [3.4.7](#), [3.4.7](#), [3.5.11](#), [3.5.11](#), [12.2.1](#), [12.3.3](#)  
subsampling, [12.8.5](#)  
variance matrix, [3.5.11](#), [3.5.11](#)  
variance of standard error, [12.4.3](#)  
wild bootstrap, [12.6](#), [12.6.5](#), [12.8.4](#)  
wild bootstrap for IV, [7.8](#), [12.6.5](#), [12.6.5](#)  
wild cluster bootstrap, [12.6.1](#), [12.6.5](#)  
bootstrap prefix, [12.4.1](#)  
`boottest` command, [3.4.6](#), [6.4.6](#), [8.2.3](#), [12.6.2](#)  
box-and-whisker plot, [2.6.2](#)  
`boxcox` command, [3.7.3](#)  
`bsample` command, [12.7.1](#)  
`bspline` command, [14.5.3](#)  
`bsqreg` command, [15.2.4](#)

by prefix, [1.3.3](#), [2.4.7](#)

bysort prefix, [2.4.7](#)

## C

capture command, [A.2.2](#)

case sensitivity, [1.3.5](#)

cd command, [1.4.1](#)

centile command, [3.2.4](#), [15.3.1](#)

chi2gof command, [11.9.3](#)

chi-squared goodness-of-fit test, *see* [specification tests](#)

Cholesky decomposition, [5.4.5](#)

ci command, [11.7.2](#)

ciwidth onemean command, [11.8.5](#)

clear command, [1.7.3](#), [2.3.1](#)

clusteff command, [6.4.6](#)

cluster analysis, [6.4](#)

cluster command, [6.4](#)

clustered data, [6.4](#), [6.7.8](#), [13.9](#), [13.9.5](#)

correlated random-effects model, [6.6.5](#), [13.9.5](#)

few clusters, [6.4.6](#), [6.4.6](#), [12.6.1](#), [12.6.4](#)

FGLS example, [6.5](#), [6.5.5](#)

fixed-effects estimator, [6.6.2](#), [6.6.2](#)

fixed-effects model, [6.6](#), [6.6.6](#)

hierarchical models, [6.7.7](#), [6.7.7](#)

influential clusters, [6.4.6](#)

linear mixed models, [6.7](#), [6.7.8](#)

nonlinear example, [13.9](#), [13.9.5](#)

nonlinear FE model example, [13.9.4](#), [13.9.5](#)

nonlinear RE model example, [13.9.3](#), [13.9.3](#)

OLS example, [6.4](#), [6.4.6](#)

population-averaged model example, [6.5.2](#), [6.5.2](#), [13.9.2](#), [13.9.2](#)

random intercept model, [6.7.4](#)

random-effects estimator, [6.5.4](#), [6.5.5](#)

random-effects model, [6.5.3](#), [6.5.5](#), [6.7.4](#)

survey methods, [6.9](#), [6.9.3](#)

two-way fixed effects, [6.6.6](#), [6.6.6](#), [8.7.4](#)

two-way random effects, [6.7.8](#)

cluster-robust inference

- cluster pairs bootstrap, [3.5.11](#), [3.5.11](#), [12.3.1](#), [12.3.5](#), [12.3.5](#), [12.4.1](#)
  - dyad-robust, [6.4.5](#)
  - few clusters, [6.2.4](#), [6.4.6](#), [6.4.6](#), [11.3.2](#), [12.6.3](#), [12.6.3](#), [12.6.4](#), [12.6.4](#)
  - for FGLS, [6.2.3](#), [6.2.4](#)
  - for mixed estimator, [6.7.5](#), [6.7.6](#)
  - for OLS, [3.4.6](#), [3.5.10](#), [6.4.2](#)
  - for panel data, [8.2.3](#), [8.2.3](#)
  - limitations, [6.2.4](#)
  - power, [11.8.4](#), [11.8.4](#)
  - two-way robust, [6.4.4](#), [6.4.4](#)
  - variance inflation, [3.4.6](#), [6.4.3](#)
  - wild cluster bootstrap, [12.6.1](#), [12.6.4](#)
- cluster-robust standard errors, *see also* [variance-covariance matrix](#)
- `cmdlog` command, [1.4.3](#)
- coefficient interpretation, *see also* [marginal effects](#)
  - in single-index models, [10.4.6](#), [13.7.3](#)
  - marginal effects, [10.4.2](#)
- `coeflegend` option, [11.3.2](#)
- `collapse` command, [2.5.4](#)
- `collect` command, [3.5.7](#)
- command prefix, [1.3.3](#)
- comma-separated values data, [2.2.1](#), [2.3.5](#)
- comments in Stata commands, [1.4.5](#)
- common correlated estimator, [9.5.8](#)
- community-contributed commands, [1.12](#)
- complier, [7.4.11](#)
- `compress` command, [2.2.2](#)
- `condivreg` command, [7.7.4](#)
- confidence intervals, [11.3](#)
  - asymmetric, [11.3.12](#)
  - asymptotic refinement, [12.3.6](#), [12.5](#), [12.6.5](#)
  - bootstrap confidence intervals, [12.3.6](#), [12.3.8](#)
  - from invert test statistic, [7.7.3](#), [11.3.13](#), [11.3.13](#)
  - percentile-*t* confidence interval, [12.5.3](#)
  - Wald confidence intervals, [11.3.9](#), [11.3.11](#)
  - with desired precision, [11.8.5](#), [11.8.5](#)

constraint command, [3.5.4](#), [6.8.6](#)  
constraints() option, [3.5.4](#)  
continue command, [1.8.4](#)  
control function estimator, [7.4.7](#)  
    standard errors, [13.3.11](#), [13.3.11](#)  
correlate command, [3.5.1](#), [8.3.9](#)  
correlation coefficient, [3.5.1](#)  
    autocorrelations for panel data, [8.3.9](#)  
    intraclass, [8.3.10](#)  
    intracluster, [6.4.3](#)  
count command, [2.4.3](#)  
count-data models, [13.2.2](#), [13.3.2](#)  
    clustered data, [13.9](#), [13.9.5](#)  
    Poisson introduction, [13.2.2](#), [13.3.2](#)  
creturn command, [1.4.6](#)  
critical values, [11.2](#), [11.2.5](#)  
    chi-squared versus  $F$ , [11.2.2](#)  
    computation, [11.2](#), [11.2.5](#)  
    standard normal versus  $t$ , [11.2.1](#), [11.2.5](#)  
    used in Stata commands, [11.2.5](#)  
cubic splines, [14.5.2](#)

## D

data frames, [2.5.3](#)  
data management, [2](#), [2.5.7](#)  
    appending datasets, [2.5.7](#)  
    collapsing and expanding data, [2.5.4](#)  
    common pitfalls, [2.3.9](#)  
    data frames, [2.5.3](#)  
    data types, [2.2](#), [2.2.4](#)  
    demeaned variables, [2.4.7](#)  
    dictionary file, [2.3.8](#)  
    exporting data, [2.4.8](#)  
    importing data, [2.3.4](#)  
    imputing missing values, [2.4.6](#)  
    indicator variables, [2.4.7](#), [2.4.7](#)  
    inputting data, [2.3](#), [2.3.10](#)

interacted variables, [1.3.4](#), [1.3.4](#), [2.4.7](#)  
labeling variables, [2.4.2](#)  
long form, [2.5.5](#), [8.10](#), [8.10.4](#)  
manipulating datasets, [2.5](#), [2.5.7](#)  
merging datasets, [2.5.6](#), [2.5.6](#)  
missing values, [2.4.5](#)  
naming variables, [2.4.2](#)  
ordering data, [2.5.1](#)  
outputting data, [2.3.10](#), [2.4.8](#)  
preserving and restoring data, [2.5.2](#)  
PSID example, [2.4.1](#)  
saving data, [2.4.8](#)  
selecting sample, [2.4.9](#)  
special data formats, [2.3.4](#)  
transforming data, [2.4.7](#), [2.4.7](#)  
wide form, [2.5.5](#), [8.10](#), [8.10.4](#)  
data summary examples, [3.2.3](#)  
    for cross-sectional data, [3.2](#), [3.2.8](#)  
    for panel data, [8.3](#), [8.3.5](#)  
data transformations, [2.4.7](#), [2.4.7](#), [3.3](#), [3.3](#), [3.4.8](#), [3.4.8](#)  
    regression in logs, [3.4.8](#)  
    retransformation, [3.4.8](#)  
data-generating process, [5.1](#)  
dataset description  
    bootstrap example, [12.3.3](#)  
    cigarette consumption panel in wide form, [8.10.1](#)  
    cigarette sales long panel example, [9.5.1](#)  
    CPS few clusters example, [12.6.3](#)  
    Medical Expenditure Panel Survey  
        doctor-visits young, [10.2.1](#)  
        drug expenditures for IV, [7.4.2](#)  
        for SUR example, [6.8.3](#)  
        medical expenditures, [3.2.1](#)  
Medicine in Australia:  
    Balancing Employment and Life doctor earnings for FMM example,  
[14.3](#)

## Balancing Employment and Life earnings for regression decomposition, [4.6.2](#)

NSW Panel Survey of Income Dynamics earnings diff-in-diff, [4.8.1](#)

Panel Study of Income Dynamics short panel wage data, [8.3.1](#)

PSID earnings data management example, [2.4.1](#)

the second National Health and Nutrition Examination Survey survey design data, [6.9.1](#)

Vietnam clustered village data, [6.4.1](#)

`date()` function, [2.4.10](#)

`decode` command, [2.4.7](#)

`delimit` command, [1.4.5](#)

`delta` method, [11.3.5](#), [11.3.12](#)

`describe` command, [2.4.1](#), [3.2.2](#)

descriptive statistics, [3.2.3](#), [3.2.4](#)

`destring` command, [2.2.3](#)

deviance residual, [13.8.3](#)

deviance statistic, [13.8.3](#)

`didregress` command, [4.8.4](#)

difference in differences, [4.8](#), [4.8.4](#)

average treatment effect on the treated, [4.8.4](#)

before–after comparison, [4.8.2](#)

OLS regression computation, [4.8.4](#)

parallel trends assumption, [4.8](#)

treatment–control comparison, [4.8.3](#)

difference in means test, [3.5.12](#)

`display` command, [1.5.1](#)

`do` command, [1.4.2](#)

do-files, [1.4](#), [1.4.6](#)

template do-file, [1.11](#)

`drawnorm` command, [5.4.5](#)

`drop` command, [2.4.9](#)

dyad–robust standard errors, [6.4.5](#), [6.4.5](#)

## E

e-class commands, [1.6.2](#), [A.2.6](#)

`egen` command, [2.4.7](#)

elasticities

in linear regression model, [4.5.7](#)  
in nonlinear models, [13.7.14](#), [13.7.15](#)

encode command, [2.4.7](#)

endogeneity test, *see* [specification tests](#)

endogenous regressors, *see also* [instrumental variables](#)

- definition, [7.3.1](#)
- dynamic panel model, [9.4](#), [9.4.10](#)
- dynamic panel systems, [9.4.10](#)
- fixed-effects model, [8.2.2](#), [8.5.1](#)
- Hausman test, [11.9.5](#), [12.4.6](#), [12.4.6](#)
- instrumental-variables regression, [7.3.3](#)
- panel IV, [9.2](#), [9.4.10](#)
- simulation example, [5.6.5](#)

ereturn command, [7.4.9](#), [7.9.1](#)

ereturn

- command, [A.1.7](#)
- display command, [3.9](#)
- list command, [1.6.2](#)

error messages, [1.3.7](#), [A.3.2](#)

estat

- abond command, [9.4.6](#)
- bootstrap command, [12.3.7](#)
- commands, [3.5.3](#), [11.5.2](#)
- endogenous command, [7.4.6](#)
- firststage command, [7.6.5](#)
- gof command, [11.9.3](#)
- hettest command, [3.7.4](#), [6.3.3](#)
- ic command, [13.8.2](#)
- imtest command, [3.7.5](#), [11.9.2](#)
- lcmean command, [14.3.3](#)
- lcprob command, [14.3.6](#)
- overid command, [7.4.8](#), [11.9.4](#)
- ovtest command, [3.7.2](#)
- sargan command, [9.4.6](#)
- vce command, [11.3.2](#)

estimates

selected command, [3.5.6](#)

`stats` command, [3.5.6](#)  
`store` command, [3.5.6](#)  
`table` command, [3.5.6](#)

estimation commands

linear panel summary, [8.2.5](#)  
nonlinear cross-sectional summary, [13.1](#)  
`estout` command, [3.5.7](#)  
`eststo` command, [3.5.7](#)  
`esttab` command, [3.5.7](#)  
`etable` command, [3.5.7](#)  
`etregress` command, [7.4.10](#)  
`expand` command, [2.5.4](#)  
`export` command, [2.3.10](#)

## F

factor variables, [1.3.4](#), [1.3.4](#), [2.4.7](#), [2.4.7](#), [3.5.8](#), [3.5.9](#), [13.7.11](#), [13.7.13](#)  
factor-variable notation, [3.5.9](#)  
    marginal effects, [4.5.4](#)  
feasible generalized least squares, *see* [generalized least squares](#)  
finite mixture models, [14](#), [14.3.10](#)  
    computational method, [14.2.6](#)  
    definition, [14.2.2](#)  
    linear regression example, [14.2.3](#), [14.2.3](#)  
    log-linear regression example, [14.3](#), [14.3.10](#)  
    marginal effects, [14.3.5](#)  
    mixture of normals, [14.2.1](#), [14.2.1](#), [14.2.5](#)  
    model selection, [14.3.8](#), [14.3.8](#)  
    modeling considerations, [14.2.4](#)  
    multimodality, [14.3.1](#)  
    predict component means, [14.3.3](#)  
    predicted class posterior probabilities, [14.3.7](#)  
    predicted class probabilities, [14.3.6](#)  
    test of coefficient equality, [14.3.9](#)  
    varying mixture probabilities, [14.3.10](#)  
fixed effects, *see* clustered data; panel data  
floating-point data, [2.2.2](#)  
`fmm` prefix, [14.2.5](#)

`foreach` command, [1.8.1](#), [13.7.10](#)  
`format` command, [2.4.3](#)  
formats to display data, [2.2.4](#), [2.4.3](#)  
`forvalues` command, [1.8.2](#), [5.3.5](#), [8.3.9](#)  
`fp` prefix, [14.4.2](#)  
`frame` command, [2.5.3](#)  
`frmttable` command, [3.5.7](#)  
`fsum` command, [3.2.4](#)

## G

`gam` command, [14.5.3](#)  
Gauss–Hermite quadrature, [5.5.1](#), [13.9.3](#)  
generalized additive model, [14.7](#)  
generalized estimating equations, [13.9.2](#), [13.9.2](#)  
generalized least squares, [6](#), [6.11](#)

- cluster-robust variance matrix, [6.2.3](#)
- definition, [6.2.2](#)
- efficient estimation, [6.2.2](#)
- for heteroskedastic errors, [6.2.1](#), [6.3.4](#)
- for SUR model, [6.8.1](#), [6.8.6](#)
- leading examples, [6.2.5](#)
- robust standard errors, [6.2.3](#), [6.2.4](#)
- robust variance matrix, [6.2.3](#)
- seemingly unrelated regressions, [6.8](#), [6.8.7](#)
- WLS estimator, [6.2.3](#), [6.3.5](#)
- working matrix, [6.2.3](#)

generalized linear models, [13.3.7](#), [13.3.8](#)

- definition, [13.3.7](#)
- link function, [13.3.8](#)
- Poisson example, [13.3.8](#)

generalized method of moments, *see also* [instrumental variables](#)

- definition, [13.3.9](#), [13.3.11](#)
- for two-step estimators, [13.3.11](#), [13.3.11](#)
- nonlinear example, [13.3.10](#)
- panel Arellano-Bond estimator, [9.4.3](#)

`generate` command, [2.4.7](#)  
`gettoken` command, [A.2.7](#)

Gibbs sampler simulation example, [5.4.6](#)

GLM, *see* [generalized linear models](#)

`glm` command, [13.3.8](#)

`global` command, [1.7.1](#)

global macros, [1.7.1](#), [1.7.1](#)

`gmm` command, [13.3.10](#)

goodness-of-fit measures, [13.8.1](#), [13.8.1](#)

graph

`box` command, [2.6.2](#)

`combine` command, [2.6.1](#), [3.2.8](#), [6.3.3](#)

`export` command, [2.6.1](#)

`matrix` command, [2.6.7](#)

`save` command, [2.6.1](#)

`twoway` command, [2.6.1](#)

graphs, [2.5.7](#), [2.6.7](#)

  box-and-whisker plot, [2.6.2](#)

  combining graphs, [2.6.1](#), [6.3.3](#)

  exporting, [2.6.1](#)

  graph formats, [2.6.1](#)

  graphing commands, [2.6.1](#)

  histogram, [2.6.3](#)

  kernel density plot, [2.6.4](#), [2.6.4](#), [3.2.8](#), [11.2.3](#)

  line plot, [2.6.5](#)

  multiple scatterplots, [2.6.7](#)

  nonparametric regression, [2.6.6](#), [2.6.6](#), [2.6.6](#), [14.6.4](#)

  of known density function, [11.2.3](#)

  of predictive margins, [4.4.6](#)

  outlier detection, [3.6.2](#)

  panel-data plots, [8.3.5](#)

  quantile plot, [15.3.1](#)

  residual plots, [3.6.2](#), [6.3.3](#)

  scatterplot, [2.6.5](#)

  Stata Graph Editor, [2.6.1](#)

`grqreg` command, [15.3.10](#)

## H

`halton()` Mata function, [5.5.4](#)

Halton sequences, [5.5.4](#)  
Hammersley sequences, [5.5.4](#)  
hausman command, [7.4.6](#), [8.8.5](#), [11.9.5](#)  
Hausman test, *see* [specification tests](#)  
Hausman-Taylor estimator, [9.3](#), [9.3.3](#)  
help  
    command, [1.2.3](#)  
    contents command, [1.2.3](#)  
    mata command, [B.1.5](#)  
heteroskedastic errors, feasible GLS, [6.2.1](#)  
heteroskedasticity test, [3.7.4](#), [6.3.3](#), [15.3.8](#)  
heteroskedastic–robust, *see* [variance-covariance matrix](#)  
hierarchical models, [6.7.7](#), [6.7.7](#)  
histogram command, [2.6.3](#)  
hypothesis tests, [11](#), [11.12](#)  
    asymptotic refinement, [12.2.3](#), [12.5](#), [12.6.5](#)  
    auxiliary regression computation, [11.5.3](#), [11.5.3](#)  
    Benjamini–Hochberg procedure, [11.6.4](#)  
    Bonferroni correction, [11.6.1](#)  
    bootstrap methods, [12](#), [12.11](#)  
    chi-squared versus  $F$ , [11.2.2](#), [11.2.5](#)  
    cross-equation restrictions, [6.8.5](#)  
    delta method, [11.3.5](#), [11.3.12](#)  
    effect size, [11.8](#)  
    false discovery proportion, [11.6.4](#)  
    false discovery rate, [11.6.4](#)  
    familywise error rate, [11.6.1](#), [11.6.1](#)  
    few clusters, [6.4.6](#), [6.4.6](#), [11.3.2](#)  
    Hochberg’s step-up procedure, [11.6.1](#)  
    Holm’s step-down procedure, [11.6.1](#)  
    invariance under transformation, [11.4.1](#)  
    invert to give confidence interval, [7.7.3](#), [11.3.13](#), [11.3.13](#)  
    Lagrange multiplier tests, [11.5](#), [11.5.3](#)  
    likelihood-ratio tests, [11.4](#), [11.4.4](#)  
    linear hypotheses, [11.3.1](#), [11.3.1](#)  
    linear regression model, [3.5.5](#)  
    minimum effect size, [11.8.1](#)

minimum sample size, [11.8.1](#)  
multiple outcomes, [11.6.3](#)  
multiple testing, [11.6](#) , [11.6.5](#)  
nonlinear hypotheses, [11.3.5](#) , [11.3.12](#)  
on difference in means, [3.5.12](#)  
on population mean, [3.2.7](#) , [3.5.12](#)  
one-sided tests, [11.3.4](#)  
percentile-*t* method, [12.5.1](#) , [12.6.5](#)  
permutation tests, [11.10](#) , [11.10](#)  
placebo test, [11.7.3](#)  
power, [5.6.3](#) , [11.7.4](#) , [11.7.5](#) , [11.8](#) , [11.8.4](#)  
power in more than one direction, [3.7.6](#) , [3.7.6](#)  
power with clustering, [11.8.4](#) , [11.8.4](#)  
pretest bias, [11.3.8](#)  
randomization tests, [11.10](#) , [12.6.1](#)  
score tests, [11.5](#) , [11.5.3](#)  
Šidák correction, [11.6.1](#)  
size, [5.6.2](#) , [11.7.2](#) , [11.7.3](#)  
standard normal versus *t*, [11.2.1](#) , [11.2.5](#)  
subgroup analysis, [11.6.2](#) , [11.6.2](#)  
tests at the boundary, [11.4.4](#)  
Wald test examples, [11.3.3](#) , [11.3.3](#) , [11.3.6](#)  
Wald tests, [11.3](#) , [11.3.13](#)  
with weak instruments, [7.7](#) , [7.8](#)

## I

`idcluster()` option, [12.3.5](#)  
`if` qualifier, [1.3.1](#)  
`import` command, [2.3.4](#)  
`in` qualifier, [1.3.1](#)  
indicator variables, [2.4.7](#) , [2.4.7](#)  
`infile` command, [2.3.6](#)  
`infix` command, [2.3.7](#)  
influential observations, [3.6.3](#) , [6.4.6](#) , [7.8](#)  
information criteria, [13.8.2](#) , [13.8.2](#) , [14.3.8](#)  
information matrix test, *see* [specification tests](#)  
`input` command, [2.3.3](#)

instrumental variables, [7](#), [7.12](#)  
2SLS bias, [7.5.7](#)  
2SLS estimator, [7.3.3](#), [7.4.1](#), [7.5](#), [7.8](#), [9.4.3](#)  
2SLS example, [7.4.4](#)  
3SLS estimator, [7.10](#), [7.10](#)  
3SLS example, [7.10](#)  
alternative estimators, [7.9](#), [7.9.4](#)  
Anderson–Rubin test, [7.7.2](#), [7.7.3](#), [7.7.4](#), [12.6.5](#)  
basic theory, [7.3.1](#), [7.3.6](#)  
binary endogenous regressor, [7.4.10](#), [7.4.10](#)  
bootstrap, [7.8](#)  
concentration parameter, [7.5.7](#)  
conditional LR tests, [7.7.4](#), [7.7.4](#)  
control function estimator, [7.4.7](#), [13.3.11](#), [13.3.11](#)  
Cragg–Donald minimum eigenvalue statistic, [7.6.3](#)  
effective  $F$  statistic, [7.6.4](#)  
example, [7.4](#), [7.4.11](#)  
finite-sample properties, [7.5.2](#), [7.5.8](#), [7.8](#), [7.8](#)  
first-stage equation, [7.3.2](#)  
first-stage  $F$  statistic, [7.5.6](#), [7.5.6](#), [7.6.6](#)  
GMM estimator, [7.3.3](#)  
Hansen test, [7.4.8](#)  
inconsistency of OLS, [7.3.1](#)  
IV estimator, [7.3.3](#)  
jackknife IV estimator, [7.9.2](#)  
just-identified model, [7.3.3](#), [7.4.4](#), [7.6.6](#)  
 $k$ -class estimator, [7.9.1](#)  
LATE, *see* [LATE](#)  
minimum distance-based tests, [7.7.5](#), [7.8](#)  
nonlinear IV, [13.3.10](#)  
optimal GMM estimator, [7.3.3](#), [7.4.1](#)  
optimal GMM example, [7.4.5](#)  
overidentified model, [7.3.3](#), [7.4.5](#), [7.6.8](#)  
panel IV estimator, [9.2](#), [9.4.10](#)  
panel systems IV estimator, [9.4.10](#)  
partial  $R^2$ , [7.6.2](#), [7.6.6](#)  
reduced form, [7.2.2](#)

relative asymptotic bias test, [7.6.4](#), [7.6.7](#)  
relative bias test, [7.6.4](#)  
robust standard errors, [7.3.6](#)  
Sargan test, [7.4.8](#), [9.4.6](#)  
sensitivity to instrument choice, [7.6.11](#)  
simple example, [7.3.1](#)  
simultaneous equations model, [7.2](#), [7.2.5](#)  
structural equation, [7.3.2](#)  
structural model approach, [7.2.1](#), [7.2.5](#)  
test of endogeneity, [7.4.6](#), [7.4.7](#)  
test of overidentifying restrictions, [7.4.8](#), [7.4.8](#)  
test of underidentification, [7.6.3](#)  
test size distortion, [7.5.8](#)  
two-sample 2SLS estimator, [7.9.4](#)  
underidentified model, [7.3.3](#)  
valid instruments, [7.3.4](#)  
Wald size-distortion test, [7.6.4](#)  
weak instruments, [7.3.5](#), [7.3.5](#), [7.5](#), [7.8](#)  
    asymptotics, [7.7.4](#), [7.7.4](#)  
    diagnostics, [7.6](#), [7.6.2](#)  
    inference, [7.7](#), [7.8](#)  
    simulation, [7.5.3](#), [7.5.5](#)  
    tests, [7.6.4](#), [7.6.10](#)  
wild bootstrap for IV, [12.6.5](#), [12.6.5](#)  
integral computation, [5.5](#), [5.5.4](#)  
interacted regressors, [3.5.8](#), [3.5.9](#), [13.7.12](#), [13.7.13](#)  
interacted variables, [1.3.4](#), [1.3.4](#), [2.4.7](#)  
interactive use of Mata, [B.1.4](#)  
interactive use of Stata, [1.1](#), [1.1](#)  
intraclass correlation, [8.3.10](#)  
intracluster correlation coefficient, [6.4.3](#)  
inverse-probability weighting, [3.8.3](#)  
`invnormal()` function, [5.2.2](#)  
`ipolate` command, [2.4.6](#)  
`iqreg` command, [15.2.4](#)  
iterative methods  
    evaluator types, [C.1.1](#), [C.2.1](#)

linear program simplex method, [15.2.2](#)  
Mata optimization, [C.1](#)  
`moptimize()` Mata function, [C.1](#), [C.1.6](#)  
optimization in Mata, [C.1](#), [C.3](#)  
`optimize()` Mata function, [C.2](#), [C.3](#)  
Poisson example in Mata, [C.2.3](#), [C.2.3](#)  
`ivreg2` command, [7.6.9](#)  
`ivregress` command, [7.4.1](#)

## J

jackknife, *see* [variance-covariance matrix](#)  
jackknife IV estimator, [7.9.2](#)  
`jackknife` prefix, [12.9.2](#)  
`javacall` command, [1.9](#)  
`jive` command, [7.9.2](#)

## K

$k$ -class estimator, [7.9.1](#)  
`kdensity` command, [2.6.4](#), [3.2.8](#)  
`keep` command, [2.4.9](#)  
kernel density plot, [2.6.4](#), [2.6.4](#)  
`knnreg` command, [14.6.2](#)  
kurtosis measure, [3.2.4](#)

## L

`label` command, [2.4.2](#)  
Lagrange multiplier test, *see* [hypothesis tests](#)  
LATE, [7.6.11](#)  
    estimator, [7.4.11](#)  
latent class models, *see* [finite mixture models](#)  
least-absolute-deviations regression, [3.6.5](#), [3.6.6](#), [15.2.2](#)  
least-squares dummy-variable estimator, [6.6.4](#), [8.5.4](#), [8.5.4](#)  
LEF, *see* [linear exponential family](#)  
leverage, [6.4.6](#), [7.8](#)  
likelihood-ratio test, *see* [hypothesis tests](#)  
LIML estimator, [7.9.1](#)  
`lincom` command, [11.3.10](#)

linear exponential family, [13.3.1](#)  
definition, [13.3.7](#)  
examples, [13.3.1](#)  
Poisson example, [13.3.7](#)

linear mixed models, [6.7](#), [6.7.8](#), [8.2.2](#)  
cluster-robust standard errors, [6.7.5](#), [6.7.6](#)

linear probability model, [10.4.7](#)

linear regression model, [3.4](#), [3.11](#), [4](#), [4.10](#)  
basic results, [3.4.1](#), [3.4.7](#)  
bootstrap variance matrix, [3.4.7](#), [3.5.11](#), [3.5.11](#)  
Box-Cox transformation, [3.7.3](#)  
cluster FE example, [6.6](#), [6.6.6](#)  
cluster FGLS example, [6.5](#), [6.5.5](#)  
cluster OLS example, [6.4](#), [6.4.6](#)  
cluster PA example, [6.5.2](#), [6.5.2](#)  
cluster RE example, [6.5.3](#), [6.5.5](#)  
cluster-robust variance, [3.4.6](#), [3.5.10](#)  
constraints on parameters, [3.5.4](#), [6.8.6](#)  
decomposition analysis, [4.6](#), [4.7.1](#)  
default standard errors, [3.4.4](#)  
difference in differences, [4.8](#), [4.8.4](#)  
elasticities, [4.5.7](#)  
endogenous regressors, [5.6.5](#), [7.1](#), [7.12](#)  
for one-sample  $t$  test, [3.5.12](#)  
for two-sample  $t$  test, [3.5.12](#)  
generalized least squares, [6](#), [6.11](#)  
heteroskedastic errors FGLS, [6.3.4](#)  
heteroskedastic–robust variance, [3.4.5](#)  
hypothesis tests, [3.5.5](#)  
influential observations, [3.6.3](#)  
instrumental variables, [7.1](#), [7.12](#)  
log versus linear regression, [3.4.8](#)  
marginal effects, [4.5](#), [4.5.8](#)  
Mata code for OLS, [3.9](#), [3.9](#)  
measurement-error example, [5.6.4](#)  
omnibus test, [3.7.5](#)  
panel-data basics, [8](#), [8.12](#)

panel-data extensions, [9](#), [9.7](#)  
prediction, [4.2](#), [4.4.8](#)  
prediction in logs, [4.2.3](#)  
prediction of actual value, [4.3.1](#)  
prediction of conditional mean, [4.3.1](#)  
prediction with binary regressor, [4.2.4](#)  
regression in logs, [3.4.8](#)  
residual analysis, [3.6.2](#), [3.6.2](#)  
retransformation problem, [4.2.3](#), [4.2.3](#)  
sampling weights, [3.8](#), [3.8.4](#)  
simulations to verify properties, [5.6](#), [5.6.5](#)  
specification analysis, [3.6](#), [3.6.6](#)  
specification tests, [3.7](#), [3.7.6](#)  
standard error estimation, [3.4.4](#), [3.4.7](#)  
survey data, [6.9](#), [6.9.3](#)  
systems of equations, [6.8](#), [6.8.7](#)  
test against log-linear, [3.7.3](#)  
test interpretation, [3.7.6](#), [3.7.6](#)  
test of heteroskedasticity, [3.7.4](#), [6.3.3](#)  
test of normality, [3.7.5](#)  
test of omitted variables, [3.7.1](#)  
transform dependent variable, [3.3](#), [3.3](#)  
two-stage least squares, [7.3.3](#)  
variance matrix estimation, [3.4.4](#), [3.4.7](#)  
weighted marginal effects, [3.8.4](#)  
weighted prediction, [3.8.4](#)  
weighted regression, [3.8.3](#), [6.9.3](#)  
link function, [13.3.8](#)  
`linktest` command, [3.7.2](#)  
`list` command, [2.3.3](#), [2.4.3](#)  
`local` command, [1.7.2](#)  
local constant regression, [2.6.6](#), [14.6.3](#)  
local linear regression, [2.6.6](#), [14.6.3](#)  
local macros, [1.7.2](#), [1.7.2](#), [1.8.3](#), [A.2.3](#), [A.2.4](#), [A.2.7](#)  
local polynomial regression, [2.6.6](#), [14.6.3](#), [14.6.4](#)  
`log` command, [1.4.3](#)  
log file, [1.4.3](#)

logical operators, [1.3.6](#)  
`logit` command, [10.5](#)  
logit model  
    brief summary, [10.2.2](#)  
log-linear regression, [3.4.8](#)  
    retransformation problem, [4.2.3](#)  
    test against linear, [3.7.3](#)  
lognormal data  
    linear regression model, [3.4.8](#)  
`loneway` command, [6.4.3](#)  
long command lines, [1.4.5](#)  
long-form data, [2.5.5](#), [8.10](#), [8.10.4](#)  
longitudinal data, *see* [panel data](#)  
looping commands, [1.8](#), [1.8.4](#)  
    `foreach`, [1.8.1](#)  
    `forvalues`, [1.8.2](#)  
    `while`, [1.8.3](#)  
`lowess` command, [2.6.6](#), [14.6.5](#)  
lowess regression, [2.6.6](#), [14.6.5](#)  
`lpoly` command, [2.6.6](#), [14.6.4](#)  
`lrtest` command, [11.4.2](#)

## M

macros, [1.7](#), [1.7.3](#)  
    compared with scalars, [1.7.3](#)  
    global, [1.7.1](#), [1.7.1](#)  
    local, [1.7.2](#), [1.7.2](#)  
marginal effects, [4.5](#), [4.5.6](#), [10.4](#), [10.4.7](#), [13.7](#), [13.7.16](#)  
    at representative value, [4.5.2](#), [4.5.6](#), [10.4.2](#), [13.7.2](#), [13.7.7](#), [13.7.7](#)  
    at the mean, [4.5.2](#), [10.4.2](#), [13.7.2](#), [13.7.6](#), [13.7.6](#)  
    average marginal effect, [3.5.9](#), [3.5.9](#), [4.5.2](#), [4.5.4](#), [10.4.2](#), [10.4.6](#),  
[13.7.2](#), [13.7.5](#), [13.7.5](#)  
    calculus method, [4.5.1](#), [10.4.1](#), [13.7.1](#)  
    calculus versus finite difference, [13.7.5](#)  
    elasticities, [4.5.7](#), [13.7.14](#), [13.7.15](#)  
    factor-variable notation, [3.5.9](#), [4.5.4](#), [13.7.11](#), [13.7.13](#)  
    finite-difference method, [4.5.1](#), [10.4.1](#), [13.7.1](#)

in linear regression model, [4.5](#), [4.5.8](#)  
in log-linear model, [3.4.8](#)  
in nonlinear models, [10.4](#), [10.4.7](#), [13.7](#), [13.7.16](#)  
in treatment-effects models, [10.4.3](#)  
interacted regressors, [3.5.9](#), [3.5.9](#), [4.5.4](#), [13.7.12](#)  
introduction for AME, [3.5.9](#), [3.5.9](#)  
manual computation, [13.7.10](#), [13.7.10](#)  
polynomial regressors, [13.7.11](#)  
population average marginal effects, [13.7.9](#)  
predictive margins, [4.4](#), [4.4.8](#), [13.6](#), [13.6.3](#)  
semielasticities, [4.5.8](#)  
summary of types, [4.5.2](#), [10.4.2](#), [13.7.2](#)  
weighted, [3.8.4](#)  
which to use, [13.7.9](#)  
marginal treatment effects, [13.7.16](#)  
user-provided quantity, [13.7.8](#)  
margins command, [4.4.2](#), [10.4.4](#), [13.6.1](#), [13.7.5](#)  
margins,  
    contrast command, [4.4.8](#)  
    dydx() command, [3.5.9](#), [4.5.3](#), [13.7.4](#), [13.7.5](#)  
    expression() command, [13.6.1](#)  
    pwcompare command, [4.4.7](#)  
marginsplot command, [4.4.6](#)  
Markov chain Monte Carlo methods  
    draws example, [5.4.6](#), [5.4.6](#)  
Mata, [1.9](#), [B](#), [B.3.5](#), [C](#), [C.3](#)  
    ado-files using Mata, [B.3.4](#)  
    colon operators, [B.2.2](#)  
    combining matrices, [B.2.5](#)  
    commands in Mata, [B.1.1](#)  
    commands in Stata, [B.1.2](#)  
    declaration of program arguments, [B.3.5](#)  
    element-by-element operators, [B.2.2](#)  
    Gibbs sampler example, [5.4.6](#)  
    help command, [B.1.5](#)  
    identity matrix, [B.2.1](#)  
    matrix

cross products, [B.2.4](#)  
functions, [B.2.3](#)  
input, [B.2.1](#)  
inversion, [B.2.3](#)  
of constants, [B.2.1](#)  
operators, [B.2.2](#)  
subscripts, [B.2.5](#)

OLS regression example, [3.9](#), [3.9](#)  
optimization functions, [C.1](#), [C.3](#)  
optimization in Mata, [C.1](#)  
overview, [B](#)  
program example, [B.3.1](#)  
programming in Mata, [B.3](#), [B.3.5](#)  
Stata commands in Mata, [B.1.3](#)  
Stata interface functions, [B.2.1](#)  
Stata matrix from Mata matrix, [B.2.6](#)  
Stata variables from Mata matrix, [B.2.6](#)

matrices in Mata, *see* [Mata](#)  
matrices in Stata, [A.1](#), [A.1.7](#)  
    combining matrices, [A.1.3](#)  
matrix  
    cross products, [A.1.6](#)  
    functions, [A.1.5](#)  
    input, [A.1.2](#)  
    operators, [A.1.4](#)  
    subscripts, [A.1.3](#)

OLS example, [A.1.7](#)  
overview, [A.1.1](#)

matrix  
    `accum` command, [A.1.6](#)  
    `command`, [1.5.2](#), [A.1](#), [A.1](#), [A.1.7](#)  
    `define` command, [A.1.2](#)  
    `rownames` command, [A.1.2](#)  
    `vecaccum` command, [A.1.6](#)

matrix algebra definitions, [3.4.2](#)  
maximum likelihood, [13.3.1](#), [13.3.3](#)  
    definition of MLE, [13.3.1](#)

misspecified density, [13.3.1](#)  
pseudo-MLE, [13.3.1](#)  
quasi-MLE, [13.3.1](#)  
mean group estimator, [9.5.8](#)  
measurement-error example, [5.6.4](#)  
median regression, [3.6.5](#), [3.6.6](#), [15.2.2](#)  
MEM, *see* marginal effects, at the mean  
MER, *see* marginal effects, at representative value  
`merge` command, [2.5.6](#)  
`mhtexp` command, [11.6.5](#)  
missing data, [2.4.5](#), [2.4.6](#)  
missing-value codes, [2.4.5](#)  
missing-values imputation, [2.4.6](#)  
`mixed` command, [6.7.3](#), [6.7.7](#)  
mixed models  
    linear, [6.7](#), [6.7.8](#)  
    nonlinear, [13.9.3](#)  
`mkspline` command, [14.5.1](#), [14.5.2](#)  
`mlexp` command, [13.3.3](#)  
model approach versus weighted regression, [3.8.3](#)  
model diagnostics, [13.8](#), [13.8.4](#)  
model selection, [11.3.7](#), [13.8.2](#), [13.8.2](#)  
    backward selection, [11.3.7](#)  
    based on statistical significance, [11.3.7](#), [11.3.8](#)  
    finite mixture models, [14.3.8](#)  
    forward selection, [11.3.7](#)  
    information criteria, [13.8.2](#), [13.8.2](#), [14.3.8](#)  
    pretest bias, [11.3.8](#)  
    stepwise selection, [11.3.7](#)  
moment-based tests, *see* [specification tests](#)  
Monte Carlo methods, *see* [simulation](#)  
`moptimize()` Mata function, [C.1](#), [C.1.6](#)  
    method `d2`, [C.1.4](#)  
    method `gf*`, [C.1.5](#)  
    method `q0`, [C.1.6](#)  
    method `q1`, [C.1.6](#)  
Poisson example, [C.1.3](#), [C.1.6](#)

`mtest` option, [11.6.1](#), [11.6.2](#)  
`multproc` command, [11.6.1](#), [11.6.2](#)  
`mvdecode` command, [2.4.5](#)

## N

natural spline, [14.5.2](#)  
nearest-neighbors regression, [14.6.2](#)  
`nl` command, [10.6.1](#), [13.3.6](#)  
`nlcom` command, [11.3.11](#)  
nonlinear IV, [13.3.10](#)  
nonlinear least squares, [13.3.5](#), [13.3.6](#)  
    estimator definition, [10.6](#), [13.3.5](#)  
    example, [13.3.6](#)  
    introduction, [10.6](#), [10.6.2](#)  
nonlinear regression, [10](#), [10.9](#), [13](#), [13.11](#)  
    cluster FE example, [13.9.4](#), [13.9.5](#)  
    cluster PA example, [13.9.2](#), [13.9.2](#)  
    cluster RE example, [13.9.3](#), [13.9.3](#)  
    clustered data example, [13.9](#), [13.9.5](#)  
    generalized linear models, [13.3.7](#), [13.3.8](#)  
    generalized method of moments, [13.3.9](#), [13.3.11](#)  
    introduction, [10](#), [10.9](#)  
    logit as example, [10.5](#), [10.5](#)  
    marginal effects, [10.4](#), [10.4.7](#), [13.7](#), [13.7.16](#)  
    maximum likelihood, [13.3.1](#), [13.3.3](#)  
    mixed models, [13.9.3](#)  
    nonlinear least squares, [10.6](#), [10.6.2](#), [13.3.5](#), [13.3.6](#)  
    Poisson as example, [13.2](#), [13.9.5](#)  
    prediction, [13.5](#), [13.5.6](#)  
    predictive margins, [13.6](#), [13.6.3](#)  
    probit as example, [10.3](#), [10.4.7](#)  
    standard errors, [13.4](#), [13.4.9](#)  
    Stata commands summary, [13.1](#)  
    summary of methods, [13.3](#), [13.3.12](#)  
    variance estimate, [13.4](#), [13.4.9](#)  
nonparametric methods, [2.6.4](#), [2.6.6](#), [14.6](#), [14.7](#)  
    bandwidth choice, [2.6.4](#)

graphs, [2.6.6](#), [2.6.6](#), [14.6.4](#)  
kernel density estimation, [2.6.4](#), [2.6.4](#)  
kernel function, [2.6.4](#), [2.6.4](#), [2.6.6](#), [14.6.3](#)  
kernel regression, [2.6.6](#), [14.6.3](#)  
local  
    constant, [2.6.6](#), [14.6.3](#)  
    linear, [2.6.6](#), [14.6.3](#)  
    polynomial, [2.6.6](#), [14.6.3](#), [14.6.4](#)  
lowess regression, [2.6.6](#), [14.6.5](#)  
nearest neighbors, [14.6.2](#)  
plugin bandwidth, [2.6.6](#), [14.6.4](#)  
semiparametric regression, [14.7](#), [14.7](#)  
npregress kernel command, [2.6.6](#), [14.6.4](#)  
npregress series command, [14.6.6](#)

## O

oaxaca command, [4.6.2](#)  
odds ratio for logit model, [10.5](#)  
one-sample *t* test, [3.5.12](#)  
    computed by OLS regression, [3.5.12](#)  
operators, [1.3.6](#), [1.3.6](#)  
optimize() Mata function, [C.2](#), [C.3](#)  
    method gf2, [C.2.3](#)  
    Poisson example, [C.2.3](#), [C.3](#)  
order command, [2.5.1](#)  
ordinary least squares, *see* [linear regression model](#)  
orthpoly command, [14.4.2](#)  
outfile command, [2.4.8](#)  
outreg command, [3.5.7](#)  
outreg2 command, [3.5.7](#)  
outsheet command, [2.4.8](#)  
outsum command, [3.2.4](#)  
overidentifying restrictions test, *see* [specification tests](#)

## P

panel data, [8](#), [8.12](#), [9](#), [9.7](#)  
    2SLS estimator, [9.4.3](#)

Arellano-Bond estimator, [9.4](#), [9.4.10](#)  
augmented mean group estimator, [9.5.8](#)  
balanced panel, [8.2.1](#)  
basics, [8](#), [8.12](#)  
between estimator, [8.6](#), [8.6.2](#)  
between variation, [8.3.4](#), [8.8.2](#)  
cluster-robust inference, [8.2.3](#), [8.2.3](#)  
cointegration, [9.5.9](#), [9.5.9](#), [9.5.9](#)  
common correlated estimator, [9.5.8](#)  
comparison of estimators, [8.8](#), [8.8.6](#)  
correlated random-effects model, [8.2.2](#), [8.7.4](#)  
data management, [8.10](#), [8.10.4](#)  
data summary, [8.3](#), [8.3.10](#)  
dynamic  
    linear model, [9.4](#), [9.4.10](#)  
    systems model, [9.4.10](#)  
endogeneity, [8.2.2](#)  
endogenous regressors, [9.2](#), [9.4.10](#)  
exogeneity, [8.9.2](#)  
few clusters, [8.2.3](#), [12.6.1](#), [12.6.4](#)  
first-difference  
    estimator, [8.9](#), [8.9.2](#)  
    IV estimator, [9.4.2](#)  
    model, [9.4.2](#)  
fixed effects, [8.5.5](#)  
fixed versus random effects, [8.8.4](#)  
fixed-effects estimator, [8.5](#), [8.5.5](#), [9.5.5](#)  
fixed-effects model, [8.2.2](#), [8.5.1](#)  
GMM estimator, [9.4.3](#)  
Hausman test, [8.8.5](#)  
Hausman-Taylor estimator, [9.3](#), [9.3.3](#)  
heterogeneous panels, [9.5.8](#)  
individual-effects model, [8.2.2](#), [8.5.1](#), [8.7.1](#), [9.5.5](#)  
individual-invariant regressor, [8.2.1](#), [8.3.4](#), [8.6](#)  
instrumental-variables estimator, [9.2](#), [9.4.10](#)  
interactive-effects estimator, [9.5.6](#)  
least-squares dummy-variable estimator, [8.5.4](#), [8.5.4](#)

linear  
extensions, [9](#), [9.7](#)  
mixed models, [8.2.2](#)  
model overview, [8.2](#), [8.2.5](#)  
long panel, [8.2.1](#), [9.5](#), [9.5.9](#), [9.5.9](#)  
long-form data, [8.10](#), [8.10.4](#)  
mean group estimator, [9.5.8](#)  
Mundlak correction, [8.7.4](#)  
pooled estimator, [8.3.8](#), [8.3.8](#), [8.4](#), [8.4.4](#), [9.5.2](#), [9.5.4](#)  
pooled model, [8.2.2](#), [8.4.1](#)  
population-averaged estimator, [8.4](#), [8.4.4](#)  
population-averaged model, [8.2.2](#)  
prediction, [8.8.6](#)  
 $R^2$ , [8.8.2](#)  
random-coefficients model, [8.2.2](#)  
random-effects estimator, [8.7](#), [8.7.4](#), [9.5.5](#)  
random-effects model, [8.2.2](#), [8.7.1](#)  
separate regressions, [9.5.7](#), [9.5.8](#)  
short panel, [8.1](#), [9.4](#)  
spatial correlation in panel, [9.5.3](#)  
Stata linear commands summary, [8.2.5](#)  
systems IV estimator, [9.4.10](#)  
time-invariant regressor, [8.2.1](#), [8.3.4](#), [8.3.8](#), [8.5.1](#), [8.5.3](#), [8.7.3](#), [8.8.4](#),  
[8.9.1](#), [9.2.3](#), [9.3](#), [9.3.3](#), [9.4.5](#)  
time-series autocorrelations, [8.3.9](#)  
time-series operators, [8.3.9](#)  
time-series plots, [8.3.5](#)  
two-way fixed effects, [8.5.5](#), [8.7.4](#)  
two-way-effects model, [8.2.2](#), [9.5.2](#)  
unbalanced panel, [8.2.1](#), [8.3.3](#), [8.8.1](#)  
unit roots, [9.5.9](#), [9.5.9](#)  
variance components, [8.8.1](#)  
wide-form data, [8.10](#), [8.10.4](#)  
within  
estimator, [8.5](#), [8.5.5](#)  
scatterplot, [8.3.7](#)  
variation, [8.3.4](#), [8.8.2](#)

partial linear model, [14.7](#)  
percentiles, [3.2.4](#)  
permutation tests, *see* [hypothesis tests](#)  
piecewise linear regression, [14.5.1](#)  
plots, *see* [graphs](#)  
plugin command, [1.9](#)  
poisson command, [13.3.2](#)  
polynomials  
    fractional, [14.4.2](#)  
    global, [14.4.1](#), [14.4.2](#), [14.6.6](#)  
    local, [14.6.3](#), [14.6.4](#)  
    orthogonal, [14.4.2](#)  
post command, [5.3.4](#)  
postclose command, [5.3.4](#)  
postestimation commands summary, [3.5.3](#), [13.3.4](#)  
postfile command, [5.3.4](#), [11.7.4](#)  
power calculations, [11.8](#), [11.8.4](#)  
power commands, [11.8](#), [11.8.4](#)  
power onemean command, [11.8.1](#)  
predict command, [4.2.1](#), [8.8.6](#), [13.5.1](#)  
prediction, [4.2](#), [4.3.2](#), [13.5](#), [13.5.6](#)  
    at a specified value, [13.5.4](#)  
    in levels from log model, [4.2.3](#)  
    in linear mixed models, [6.7.6](#)  
    in linear panel-data model, [8.8.6](#)  
    in linear regression model, [4.2](#), [4.4.8](#)  
    in nonlinear models, [13.5](#), [13.5.6](#)  
    in sample, [4.2](#), [4.2.5](#)  
    of actual value, [4.3.1](#), [13.5.2](#)  
    of conditional mean, [4.3.1](#), [13.5.2](#)  
    of nonlinear quantity, [13.5.1](#)  
    out of sample, [4.3](#), [4.3.2](#), [13.5.3](#)  
    weighted, [3.8.4](#)  
    with binary regressor, [4.2.4](#)  
predictive margins, [4.4](#), [4.4.8](#), [13.6](#), [13.6.3](#)  
    at a specified value, [13.6.2](#)  
    at means, [13.6.2](#)

average, [13.6.1](#), [13.6.2](#)  
contrasts of predictive margins, [4.4.8](#)  
for a categorical variable, [4.4.4](#), [13.6.3](#)  
for a continuous variable, [4.4.5](#)  
in nonlinear models, [13.6](#), [13.6.3](#)  
pairwise comparisons, [4.4.7](#)  
plots of predictive margins, [4.4.6](#)  
predictive means, [4.4](#), [4.4.8](#)  
`predictnl` command, [13.5.1](#)  
prefix command, [1.3.3](#)  
`preserve` command, [2.5.2](#), [4.2.4](#), [13.5.4](#), [13.7.10](#)  
`probit` command, [10.3.1](#)  
probit model  
    brief summary, [10.2.2](#)  
program `define` command, [A.2.1](#)  
program `drop` command, [A.2.2](#)  
programs, [A.2](#), [A.4](#)  
    bootstrap example, [12.4.4](#), [12.5.2](#), [12.7.2](#)  
    central limit theorem example, [5.3.1](#), [5.3.5](#)  
    debugging, [A.3](#)  
    in Mata, [B.3](#), [B.3.5](#)  
    in Stata, [5.3.1](#), [A.2](#), [A.4](#)  
    Monte Carlo integration, [5.5.3](#), [5.5.3](#)  
    named positional arguments, [A.2.5](#)  
    OLS with chi-squared errors, [5.6.1](#), [5.6.3](#)  
    overview, [A.2](#)  
    parsing syntax, [A.2.7](#)  
    positional arguments, [A.2.3](#)  
    r-class example, [A.2.6](#)  
    temporary variables, [A.2.4](#)  
`prvalue` command, [13.7.12](#)  
 $p$ -pseudo- $R^2$ , [13.8.1](#)  
 $p$ -values, [11.2](#), [11.2.5](#)  
    bootstrap, [12.5.2](#)  
    chi-squared versus  $F$ , [11.2.2](#)  
    computation, [11.2](#), [11.2.5](#)  
    standard normal versus  $t$ , [11.2.1](#), [11.2.5](#)

used in Stata commands, [11.2.5](#)  
`pvar` command, [9.4.10](#)  
`pwcorr` command, [3.5.1](#)  
`python` command, [1.9](#)

## Q

`qnorm` command, [3.6.2](#)  
`qplot` command, [15.3.1](#)  
`qqplot` command, [3.6.2](#)  
`qreg` command, [3.6.5](#), [15.2.4](#)  
`qreg2` command, [15.3.6](#)  
quadrature methods, [5.5.1](#)  
adaptive Gauss–Hermite, [5.5.1](#)  
quantile regression, [15](#), [15.7](#)  
bootstrap robust variance, [15.2.4](#)  
censored data, [15.3.11](#)  
coefficient comparison across quantiles, [15.3.5](#), [15.3.10](#)  
coefficient interpretation, [15.3.3](#)  
computation, [15.2.2](#)  
conditional quantiles, [15.2](#), [15.5](#)  
data example, [15.3](#), [15.3.10](#)  
definition of quantiles, [15.1](#)  
endogenous data, [15.3.11](#)  
generated data example, [15.4](#), [15.4.2](#)  
loss functions, [15.2.1](#)  
marginal effects, [15.3.4](#)  
quantile estimator, [15.2.2](#)  
retransformation, [15.3.4](#)  
robust standard errors, [15.2.3](#), [15.3.6](#)  
test of coefficient equality, [15.3.9](#)  
test of heteroskedasticity, [15.3.8](#)  
treatment effect, [15.5](#), [15.5](#)  
variance matrix estimation, [15.2.3](#)  
`query` command, [1.4.6](#)

## R

$R^2$ , [13.8.1](#)

for linear panel models, [8.8.2](#)  
partial  $R^2$ , [7.6.2](#)  
pseudo- $R^2$ , [13.8.1](#)

random effects, *see* clustered data; panel data  
randomization tests, *see* [hypothesis tests](#)  
random-number generation, *see* [simulation](#)

`rbeta()` function, [5.2.3](#)  
`rbinomial()` function, [5.2.4](#)  
`rchi2()` function, [5.2.3](#)  
r-class commands, [1.6.1](#), [5.3.1](#), [A.2.6](#)  
`recode` command, [2.4.7](#)  
recursive models, [7.2.3](#)  
`reg3` command, [7.10](#)  
`rego` command, [4.7.1](#)  
`regress` command, [1.3.3](#), [1.6.2](#), [3.5.2](#), [6.6.4](#)  
regression decomposition analysis  
    Oaxaca–Blinder, [4.6.1](#), [4.6.2](#)  
    Shapley relative importance, [4.7](#), [4.7.1](#)  
regression splines, [14.5](#), [14.5.3](#), [14.6.6](#)  
    basis expansions, [14.5.3](#)  
    cubic splines, [14.5.2](#)  
    knots, [14.5.1](#)  
    natural spline, [14.5.2](#)  
    piecewise linear regression, [14.5.1](#)  
    restricted spline, [14.5.2](#)  
    smoothing splines, [14.5.3](#)  
relational operators, [1.3.6](#)  
`rename` command, [2.4.2](#)  
`replace` command, [2.4.5](#), [2.4.7](#)  
report generation, [3.10](#)  
resampling methods, *see* [bootstrap methods](#)  
`reshape` command, [2.5.5](#), [4.8.4](#), [8.10](#), [8.10.4](#)  
residuals  
    definitions of various residuals, [13.8.3](#), [13.8.3](#)  
    residual plots, [3.6.2](#)  
`restore` command, [2.5.2](#), [4.2.4](#), [13.5.4](#), [13.7.10](#)  
`return` code, [1.3.7](#), [A.3.2](#)

return command, [5.6.1](#)  
return list command, [1.6.1](#)  
rgamma() function, [5.2.3](#)  
rivtest command, [7.7.5](#)  
rnbinomial() function, [5.2.4](#)  
rnormal() function, [5.2.2](#)  
robust regression, [3.6.1](#), [3.6.1](#), [3.6.4](#), [3.6.6](#)  
    influential observations, [3.6.3](#)  
robust standard errors, *see* [variance-covariance matrix](#)  
rpoisson() function, [5.2.4](#)  
rreg command, [3.6.4](#)  
rt() function, [5.2.3](#)  
runiform() function, [5.2.1](#)  
rvfplot command, [3.6.2](#)

## S

sample command, [2.4.9](#)  
sampling weights, [3.8.1](#), [6.9](#)  
    in linear regression model, [3.8](#), [3.8.4](#), [6.9](#), [6.9.3](#)  
    weighted mean, [3.8.2](#), [6.9.2](#)  
save command, [2.4.8](#)  
saveold command, [2.4.8](#)  
scalars, [1.5.1](#), [1.7.3](#)  
    compared with macros, [1.7.3](#)  
scatterplot, [2.6.5](#)  
score test, *see* [hypothesis tests](#)  
search command, [1.2.4](#)  
seed for bootstrap, [12.2.4](#)  
seed for random-number generation, [5.2.1](#)  
seemingly unrelated estimations, [6.8.7](#)  
seemingly unrelated regression equations, [6.8](#), [6.8.7](#)  
    cross-equation restrictions test, [6.8.5](#)  
    feasible GLS estimation, [6.8.1](#), [6.8.6](#)  
    FGLS example, [6.8.3](#)  
    imposing cross-equation restrictions, [6.8.6](#)  
    robust standard errors, [6.8.4](#)  
    SUR estimator, [6.8.1](#)

SUR model, [6.8.1](#)  
test of error independence, [6.8.3](#)  
selection models  
    two-step estimation, [12.4.5](#)  
semielasticities, [13.7.14](#), [13.7.15](#)  
    in linear regression model, [4.5.8](#)  
semiparametric methods, [14.7](#), [14.7](#)  
series regression, [14.6.6](#), [14.6.6](#)  
set  
    command, [1.4.6](#)  
    dots command, [3.5.11](#)  
    iterlog command, [3.6.6](#)  
    matsize command, [6.6.4](#)  
    seed command, [5.2.1](#), [12.2.4](#), [15.2.4](#)  
simulate prefix, [5.3.2](#)  
simulation, [5](#), [5.8](#)  
    asymptotic power, [11.7.5](#)  
    bias of estimator, [5.6.2](#)  
    bias of standard error estimator, [5.6.2](#)  
    bootstrap example, [12.7.2](#)  
    central limit theorem example, [5.3](#), [5.3.5](#)  
    Cholesky decomposition, [5.4.5](#)  
    computing integrals, [5.5](#), [5.5.4](#)  
    direct transformation, [5.4.2](#)  
    distribution of sample mean, [5.3](#), [5.3.5](#)  
    draws from multivariate normal, [5.4.5](#)  
    draws from normal, [5.2.2](#)  
    draws from truncated normal, [5.4.4](#)  
    draws from uniform, [5.2.1](#)  
    draws of continuous variates, [5.2.3](#)  
    draws of discrete variates, [5.2.4](#)  
    draws using Markov chain Monte Carlo method, [5.4.6](#), [5.4.6](#)  
endogeneity example, [7.2.4](#), [7.2.5](#)  
endogenous regressors example, [5.6.5](#)  
FGLS heteroskedastic errors example, [6.3.1](#)  
Gibbs sampler example, [5.4.6](#)  
Halton sequences, [5.5.4](#)

Hammersley sequences, [5.5.4](#)  
inconsistency of estimator, [5.6.4](#)  
interpreting simulation output, [5.6.2](#)  
inverse-probability transformation, [5.4.1](#)  
linear regression example, [5.6](#) , [5.6.5](#)  
measurement-error example, [5.6.4](#)  
mixture of distributions, [5.4.3](#)  
Monte Carlo integration, [5.5.3](#) , [5.5.4](#)  
OLS with chi-squared errors, [5.6.1](#) , [5.6.3](#)  
pseudorandom numbers, [5.2](#) , [5.2.4](#)  
seed, [5.2.1](#)  
test power computation, [5.6.3](#) , [11.7.4](#) , [11.7.4](#)  
test size computation, [5.6.2](#) , [11.7.2](#) , [11.7.2](#)  
using actual data, [11.7.3](#) , [11.7.3](#)  
using postfile command, [5.3.4](#)  
using simulate prefix, [5.3.2](#) , [5.6](#)  
weak-instruments example, [7.5.3](#) , [7.5.5](#)  
simultaneous equations model, [7.2](#) , [7.2.5](#)  
3SLS estimation, [7.10](#) , [7.10](#)  
first-stage equation, [7.2.3](#)  
recursive system, [7.2.3](#)  
reduced form, [7.2.2](#)  
structural equation, [7.2.3](#)  
structural model, [7.2.1](#)  
single-index models, [14.7](#)  
coefficient interpretation, [10.4.6](#) , [13.7.3](#)  
definition, [13.7.3](#)  
skewness measure, [3.2.4](#)  
sort command, [2.5.1](#)  
specification analysis, [3.6](#) , [3.6.6](#)  
specification tests, [3.7](#) , [3.7.6](#) , [11.9](#) , [11.9.5](#) , [13.8.4](#)  
chi-squared goodness-of-fit test, [11.9.3](#)  
first-stage  $F$  statistic, [7.6.4](#) , [7.6.10](#)  
for fixed effects, [8.8.5](#) , [8.8.5](#)  
for serially uncorrelated panel errors, [9.4.6](#)  
Hausman test, [8.8.5](#) , [8.8.5](#) , [11.9.5](#) , [12.4.6](#) , [12.4.6](#)  
information matrix test, [11.9.2](#)

moment-based tests, [11.9.1](#)  
of endogeneity, [7.4.6](#), [7.4.6](#), [12.4.6](#), [12.4.6](#)  
of heteroskedasticity, [3.7.4](#), [6.3.3](#), [15.3.8](#)  
of omitted variables, [3.7.1](#)  
of overidentifying restrictions, [7.4.8](#), [7.4.8](#), [9.4.6](#), [11.9.4](#)  
of underidentification, [7.6.3](#)  
of weak instruments, [7.6.4](#), [7.6.10](#)  
panel cointegration, [9.5.9](#), [9.5.9](#)  
panel unit roots, [9.5.9](#), [9.5.9](#)  
RESET test, [3.7.2](#)  
robust tests, [7.7.5](#)  
    test of heteroskedasticity, [3.7.4](#)  
spline regression, *see* [regression splines](#)  
`spshape2dta` command, [2.3.4](#)  
`sqreg` command, [15.2.4](#)  
`st_addvar()` Mata function, [B.2.6](#)  
standard error estimation, *see* [variance-covariance matrix](#)  
standardized regression, [3.11](#)  
Stata documentation, [1.2.1](#), [1.2.4](#)  
*Stata Journal*, [1.2.2](#)  
*Stata Technical Bulletin*, [1.2.2](#)  
`stata()` Mata function, [B.1.3](#)  
`statsby` prefix, [9.5.7](#)  
`st_data()` Mata function, [B.2.1](#)  
`stepwise` command, [11.3.7](#)  
`st_matrix()` Mata function, [3.9](#), [B.2.1](#), [B.2.6](#)  
string data, [2.2.3](#), [2.3.3](#)  
`st_store()` Mata function, [B.2.6](#)  
`st_view()` Mata function, [3.9](#), [B.2.1](#)  
subsampling, *see* [bootstrap methods](#)  
`suest` command, [4.6.2](#), [6.8.7](#)  
`summarize` command, [1.3.2](#), [1.6.1](#), [3.2.3](#)  
summary statistics, *see* [data summary examples](#)  
`summcluster` package, [6.4.6](#)  
SUR model, *see* [seemingly unrelated regression equations](#)  
`sureg` command, [6.8.2](#)  
survey data, [3.8](#), [3.8.4](#), [6.9](#), [6.9.3](#)

clustering, [6.9](#)  
commands for survey data, [6.9.1](#)  
complex survey data, [6.9](#)  
stratification, [6.9](#)  
weighted mean, [3.8.2](#), [6.9.2](#)  
weighted regression, [3.8.3](#), [6.9.3](#)  
weighting, [6.9](#)  
`svy` prefix, [6.9](#)  
`svy:` `regress` command, [6.9.3](#)  
`svydescribe` command, [6.9.1](#)  
`svymean` command, [6.9.2](#)  
`svyset` command, [6.9.1](#)  
syntax, [1.3](#), [1.3.1](#)  
    basic command syntax, [1.3](#)  
    parsing syntax, [A.2.7](#)  
`syntax` command, [A.2.7](#)  
systems of equations  
    linear regressions, [6.8](#), [6.8.7](#)  
    linear simultaneous equations, [7.10](#), [7.10](#)  
`sysuse` command, [1.3.2](#)

## T

`tab2` command, [3.2.5](#)  
`table` command, [3.2.5](#), [3.5.7](#)  
tables of frequencies, [3.2.5](#), [3.2.5](#)  
tables of regression output, [3.5.6](#)  
    in Word or LaTeX, [3.10](#)  
tables of summary statistics, [3.2.6](#), [3.2.6](#)  
`tabstat` command, [3.2.6](#)  
`tabulate` command, [2.4.3](#), [2.4.7](#), [3.2.3](#)  
`tempvar` command, [A.2.4](#)  
`test` command, [3.5.5](#), [11.3.2](#)  
test of overidentifying restrictions, *see* [specification tests](#)  
`testnl` command, [11.3.6](#)  
`testparm` command, [11.3.2](#)  
tests, *see* [hypothesis tests](#)  
text data, [2.2.1](#)

three-stage least squares, [7.10](#), [7.10](#)  
time-series data, [2.4.10](#), [2.4.10](#)  
    within panel, [8.3.9](#)  
time-series operators, [8.3.9](#)  
`tokens()` Mata function, [3.9](#)  
`toString` command, [2.2.3](#)  
`trace` command, [A.3.3](#)  
treatment effects  
    difference in differences, [4.8](#), [4.8.4](#)  
    quantile regression, [15.5](#), [15.5](#)  
treatment-control comparison, [4.8.3](#)  
`tsset` command, [2.4.10](#), [5.2.1](#)  
`ttest` command, [3.2.7](#), [3.5.12](#)  
    compared with `regress`, [3.5.12](#)  
two-sample 2SLS estimator, [7.9.4](#)  
two-sample *t* test, [3.5.12](#)  
    computed by OLS regression, [3.5.12](#)  
two-stage least-squares estimator, *see* [instrumental variables](#)  
two-step estimator  
    bootstrap standard error, [12.4.5](#), [12.4.5](#)  
    stacked GMM standard error, [13.3.11](#), [13.3.11](#)  
two-way cluster-robust, [6.4.4](#), [6.4.4](#)  
`twoway` command, [2.6.5](#)  
`type` command, [1.4.1](#)

## U

`update` command, [1.1](#)  
`use` command, [2.3.2](#)

## V

variable names and labels, [2.4.2](#), [2.4.2](#)  
variance-covariance matrix, [3.4.7](#), [3.4.7](#), [13.4](#), [13.4.9](#)  
    bootstrap estimate, [3.4.7](#), [3.5.11](#), [3.5.11](#), [12.3](#), [12.3.5](#), [13.4.8](#)  
    cluster-robust, [3.4.6](#), [3.5.10](#), [6.4.2](#), [6.4.6](#), [6.4.6](#), [8.2.3](#), [8.2.3](#), [13.4.6](#)  
    default estimate, [3.4.4](#), [13.4.4](#)  
    dyad-robust, [6.4.5](#)  
    for FGLS, [6.2.3](#), [6.2.4](#)

for IV, [7.3.6](#)  
for  $m$ -estimator, [13.4.1](#)  
for nonlinear estimators, [13.4](#), [13.4.9](#)  
for OLS, [3.4.4](#), [3.4.7](#)  
for SUR, [6.8.4](#)  
for two-step estimator, [12.4.5](#), [12.4.5](#), [13.3.11](#), [13.3.11](#)  
HAC estimate, [13.4.7](#)  
heteroskedastic–robust, [3.4.5](#), [13.4.5](#)  
jackknife estimate, [12.9.1](#), [12.9.2](#), [13.4.8](#)  
outer product of the gradient, [13.4.4](#)  
robust when to use, [13.3.1](#), [13.4.2](#)  
sandwich form, [13.4.1](#)  
two-way cluster-robust, [6.4.4](#), [6.4.4](#)  
unconditional, [4.5.3](#), [13.7.4](#), [13.7.9](#)  
vce (bootstrap) option, [10.3.3](#)

VCE, *see* [variance-covariance matrix](#)  
vce() option, *see* [variance-covariance matrix](#)  
vce(bootstrap) option, [3.4.7](#), [12.3.2](#), [13.4.8](#)  
vce(cluster *clustvar*) option, [3.4.6](#), [13.4.6](#), [13.7.9](#)  
vce(hac) option, [13.4.7](#)  
vce(jackknife) option, [12.9.2](#), [13.4.8](#)  
vce(oim) option, [13.4.4](#)  
vce(opg) option, [13.4.4](#)  
vce(robust) option, [3.4.5](#), [13.4.5](#)  
vce(unconditional) option, [4.5.3](#), [13.7.4](#), [13.7.9](#), [13.10](#)  
vcemway command, [6.4.4](#)

## W

Wald

confidence intervals, *see* [confidence intervals](#)  
test, *see* [hypothesis tests](#)

weak instruments, *see* [instrumental variables](#)

weakiv command, [7.7.5](#)

weakivtest command, [7.6.7](#)

weight, [1.3.1](#)

weighted mean, [3.8.2](#)

survey data, [6.9.2](#)

weighted regression, *see* [inverse-probability weighting](#)  
survey data, [6.9.3](#)  
`while` command, [1.8.3](#)  
wide-form data, [2.5.5](#), [8.10](#), [8.10.4](#)  
wild gradient bootstrap, [15.2.3](#)  
wildcards, [1.3.5](#)

## X

`xi` prefix, [2.4.7](#)  
`xtabond` command, [9.4.3](#)  
`xtabond2` command, [9.4.9](#)  
`xtcointtest` command, [9.5.9](#)  
`xtdata` command, [8.3.7](#)  
`xtdescribe` command, [6.7](#), [8.3.3](#)  
`xtdidregress` command, [4.8.4](#)  
`xtdpd` command, [9.4.8](#)  
`xtdpdsys` command, [9.4.7](#)  
`xtgee` command, [8.4.3](#), [13.9.2](#)  
`xtgls` command, [9.5.3](#), [9.5.4](#)  
`xttaylor` command, [9.3.2](#)  
`xtivreg` command, [9.2.2](#)  
`xtline` command, [8.3.5](#)  
`xtlogitfe` command, [13.9.4](#)  
`xtmg` command, [9.5.8](#)  
`xtoverid` command, [8.8.5](#)  
`xtpcse` command, [9.5.3](#)  
`xtpmg` command, [9.5.8](#)  
`xtprobitfe` command, [13.9.4](#)  
`xtreg` commands, [6.5](#), [8.2.4](#)  
`xtreg`,  
  `be` command, [8.6.2](#)  
  `fe` command, [6.6.2](#), [8.5.2](#)  
  `mle` command, [8.7.2](#)  
  `pa` command, [6.5.2](#), [8.4.3](#)  
  `re` command, [6.5.4](#), [8.7.2](#)  
`xtregar` command, [9.5.5](#)  
`xtscc` command, [9.5.3](#)

`xtset` command, [6.5.1](#), [8.3.2](#)

`xtsum` command, [8.3.4](#)

`xttab` command, [8.3.4](#)

`xttrans` command, [8.3.4](#)

`xtunitroot` command, [9.5.9](#)